This file is a merged representation of the entire codebase, combined into a single document by Repomix.

<file_summary>
This section contains a summary of this file.

<purpose>
This file contains a packed representation of the entire repository's contents.
It is designed to be easily consumable by AI systems for analysis, code review,
or other automated processes.
</purpose>

<file_format>
The content is organized as follows:
1. This summary section
2. Repository information
3. Directory structure
4. Repository files (if enabled)
5. Multiple file entries, each consisting of:
  - File path as an attribute
  - Full contents of the file
</file_format>

<usage_guidelines>
- This file should be treated as read-only. Any changes should be made to the
  original repository files, not this packed version.
- When processing this file, use the file path to distinguish
  between different files in the repository.
- Be aware that this file may contain sensitive information. Handle it with
  the same level of security as you would the original repository.
</usage_guidelines>

<notes>
- Some files may have been excluded based on .gitignore rules and Repomix's configuration
- Binary files are not included in this packed representation. Please refer to the Repository Structure section for a complete list of file paths, including binary files
- Files matching patterns in .gitignore are excluded
- Files matching default ignore patterns are excluded
- Files are sorted by Git change count (files with more changes are at the bottom)
</notes>

</file_summary>

<directory_structure>
.repomix/bundles.json
admin/.gitignore
admin/biome.json
admin/next.config.ts
admin/package.json
admin/postcss.config.mjs
admin/proxy.ts
admin/public/BackGround.png
admin/public/file.svg
admin/public/globe.svg
admin/public/Menus/BackGround.png
admin/public/Menus/BurgerMeal.png
admin/public/Menus/Cappy.png
admin/public/Menus/ChickenAndBBQ.png
admin/public/Menus/ChickenAndMayo.png
admin/public/Menus/ChickenMeal.png
admin/public/Menus/Chips.png

admin/public/Menus/Coca-Cola.png
admin/public/Menus/Drinks.png
admin/public/Menus/Drinks2.png
admin/public/Menus/FishMeal.png
admin/public/Menus/Kota.jpg
admin/public/Menus/Kota.png
admin/public/Menus/KotaMeal.png
admin/public/Menus/Main Pizza131121.png
admin/public/Menus/Meals.png
admin/public/Menus/Menu.png
admin/public/Menus/mexicanChilli.png
admin/public/Menus/petter.png
admin/public/Menus/Pineapply.png
admin/public/next.svg
admin/public/PA_Logo.png
admin/public/vercel.svg
admin/public/window.svg
admin/README.md
admin/src/app/(auth)/layout.tsx
admin/src/app/(auth)/login/page.tsx
admin/src/app/(auth)/register/page.tsx
admin/src/app/(dashboard)/contact/page.tsx
admin/src/app/(dashboard)/dashboard/page.tsx
admin/src/app/(dashboard)/featured-items/page.tsx
admin/src/app/(dashboard)/layout.tsx
admin/src/app/(dashboard)/menu/page.tsx
admin/src/app/(dashboard)/menu/products/[id]/edit/page.tsx
admin/src/app/(dashboard)/orders/page.tsx
admin/src/app/(dashboard)/photo-bookings/page.tsx
admin/src/app/(dashboard)/profiles/page.tsx
admin/src/app/(dashboard)/settings/page.tsx
admin/src/app/(dashboard)/support/page.tsx
admin/src/app/(dashboard)/testimonials/page.tsx
admin/src/app/favicon.ico
admin/src/app/globals.css
admin/src/app/layout.tsx
admin/src/app/page.tsx
admin/src/components/auth/AuthShell.tsx
admin/src/components/auth/LoginForm.tsx
admin/src/components/auth/RegisterForm.tsx
admin/src/components/contact/ContactBoard.tsx
admin/src/components/dashboard/LowInventoryCard.tsx
admin/src/components/dashboard/RecentOrdersCard.tsx
admin/src/components/dashboard/RevenueTrendCard.tsx
admin/src/components/dashboard/StatsGrid.tsx
admin/src/components/featured-items/FeaturedItemsBoard.tsx
admin/src/components/forms/ImageUpload.tsx
admin/src/components/forms/SubmitButton.tsx
admin/src/components/layout/AdminSidebar.tsx
admin/src/components/layout/AdminTopbar.tsx
admin/src/components/layout/DashboardShell.tsx
admin/src/components/layout/nav-items.tsx
admin/src/components/layout/SidebarNav.tsx
admin/src/components/menu/CategoriesBoard.tsx
admin/src/components/menu/comments/CommentsBoard.tsx
admin/src/components/menu/favorites/FavoritesBoard.tsx
admin/src/components/menu/products/AddProductForm.tsx
admin/src/components/menu/products/AddProductPanel.tsx
admin/src/components/menu/products/ProductBoard.tsx
admin/src/components/menu/products/ProductEditForm.tsx
admin/src/components/menu/QuickEditForm.tsx
admin/src/components/orders/OrdersBoard.tsx
admin/src/components/orders/UpdateOrderStatusForm.tsx
admin/src/components/photo-bookings/PhotoBookingsBoard.tsx

admin/src/components/profiles/ProfileDetailModal.tsx
admin/src/components/profiles/ProfilesBoard.tsx
admin/src/components/shared/DeleteConfirmDialog.tsx
admin/src/components/testimonials/TestimonialsBoard.tsx
admin/src/lib/auth/actions.ts
admin/src/lib/auth/index.ts
admin/src/lib/auth/session.ts
admin/src/lib/auth/types.ts
admin/src/lib/data/categories-actions.ts
admin/src/lib/data/categories.ts
admin/src/lib/data/comments-actions.ts
admin/src/lib/data/comments.ts
admin/src/lib/data/contact.ts
admin/src/lib/data/dashboard.ts
admin/src/lib/data/favorites.ts
admin/src/lib/data/orders-actions.ts
admin/src/lib/data/orders.ts
admin/src/lib/data/photo-bookings-actions.ts
admin/src/lib/data/photo-bookings.ts
admin/src/lib/data/products-actions.ts
admin/src/lib/data/products.ts
admin/src/lib/data/profile-details.ts
admin/src/lib/data/profiles-actions.ts
admin/src/lib/data/profiles.ts
admin/src/lib/data/testimonials-actions.ts
admin/src/lib/data/testimonials.ts
admin/src/lib/database.types.ts
admin/src/lib/supabase/client.ts
admin/src/lib/supabase/config.ts
admin/src/lib/supabase/index.ts
admin/src/lib/supabase/server.ts
admin/src/lib/supabase/storage.ts
admin/src/lib/types/base.ts
admin/src/lib/types/database-records.ts
admin/src/lib/types/index.ts
admin/src/lib/types/orders.ts
admin/src/lib/types/products.ts
admin/src/lib/types/testimonials.ts
admin/src/lib/types/users.ts
admin/tsconfig.json
client/.gitignore
client/biome.json
client/next.config.ts
client/package.json
client/postcss.config.mjs
client/public/BackGround.png
client/public/Carwash.png
client/public/Central_Eatery_Logo.png
client/public/file.svg
client/public/globe.svg
client/public/Menus/BackGround.png
client/public/Menus/BurgerMeal.png
client/public/Menus/Cappy.png
client/public/Menus/ChickenAndBBQ.png
client/public/Menus/ChickenAndMayo.png
client/public/Menus/ChickenMeal.png
client/public/Menus/Chips.png
client/public/Menus/Coca-Cola.png
client/public/Menus/Drinks.png
client/public/Menus/Drinks2.png
client/public/Menus/FishMeal.png
client/public/Menus/Kota.jpg
client/public/Menus/Kota.png
client/public/Menus/KotaMeal.png

client/public/Menus/Main Pizza131121.png
client/public/Menus/Meals.png
client/public/Menus/Menu.png
client/public/Menus/mexicanChilli - Copy.png
client/public/Menus/mexicanChilli.png
client/public/Menus/petter.png
client/public/Menus/Pineapply - Copy.png
client/public/Menus/Pineapply.png
client/public/next.svg
client/public/PA_Logo.png
client/public/PhotoBoot.jpg
client/public/robots.txt
client/public/users/Abram.jpg
client/public/vercel.svg
client/public/window.svg
client/README.md
client/src/app/(auth)/confirm-email/page.tsx
client/src/app/(auth)/login/loading.tsx
client/src/app/(auth)/login/page.tsx
client/src/app/(auth)/register/loading.tsx
client/src/app/(auth)/register/page.tsx
client/src/app/about/components/CompanyDescription.tsx
client/src/app/about/components/CoreValues.tsx
client/src/app/about/components/KeyHighlights.tsx
client/src/app/about/components/KeyLeadership.tsx
client/src/app/about/components/MilestonesAndVision.tsx
client/src/app/about/components/OurCommitment.tsx
client/src/app/about/components/OurJourney.tsx
client/src/app/about/components/OurVision.tsx
client/src/app/about/components/StrengthsAndOpportunities.tsx
client/src/app/about/components/WeaknessesAndThreats.tsx
client/src/app/about/components/WhatMakesUsSpecial.tsx
client/src/app/about/loading.tsx
client/src/app/about/page.tsx
client/src/app/api/seed/route.ts
client/src/app/blog/components/BlogPostList.tsx
client/src/app/blog/loading.tsx
client/src/app/blog/page.tsx
client/src/app/contact/components/ContactInfo.tsx
client/src/app/contact/components/MapEmbed.tsx
client/src/app/contact/components/OperatingHours.tsx
client/src/app/contact/loading.tsx
client/src/app/contact/page.tsx
client/src/app/favicon.ico
client/src/app/gallery/components/ImageGallery.tsx
client/src/app/gallery/components/ImageModal.tsx
client/src/app/gallery/loading.tsx
client/src/app/gallery/page.tsx
client/src/app/globals.css
client/src/app/home-components/AboutUsSnippet.tsx
client/src/app/home-components/ContactSection.tsx
client/src/app/home-components/FeaturedItemsServices.tsx
client/src/app/home-components/FinancialSnapshotTeaser.tsx
client/src/app/home-components/HeroSection.tsx
client/src/app/home-components/KeyDifferentiators.tsx
client/src/app/home-components/KeyHighlights.tsx
client/src/app/home-components/TargetMarketCallout.tsx
client/src/app/home-components/Testimonials.tsx
client/src/app/layout.css
client/src/app/layout.tsx
client/src/app/loading.tsx
client/src/app/menu/[Products]/[product]/comments/loading.tsx
client/src/app/menu/[Products]/[product]/comments/page.tsx
client/src/app/menu/[Products]/[product]/components/AddtoCart.tsx

client/src/app/menu/[Products]/[product]/components/comments.tsx
client/src/app/menu/[Products]/[product]/components/FavoriteButton.tsx
client/src/app/menu/[Products]/[product]/components/ProductInfo.tsx
client/src/app/menu/[Products]/[product]/components/RelatedProducts.tsx
client/src/app/menu/[Products]/[product]/loading.tsx
client/src/app/menu/[Products]/[product]/page.tsx
client/src/app/menu/[Products]/components/CommentModal.tsx
client/src/app/menu/[Products]/components/CommentsButton.tsx
client/src/app/menu/[Products]/components/FilterSortBar.tsx
client/src/app/menu/[Products]/components/LikesButton.tsx
client/src/app/menu/[Products]/components/SocialButtons.tsx
client/src/app/menu/[Products]/loading.tsx
client/src/app/menu/[Products]/page.tsx
client/src/app/menu/[Products]/utils/likesUtils.ts
client/src/app/menu/cart/components/CartFooter.tsx
client/src/app/menu/cart/loading.tsx
client/src/app/menu/cart/page.tsx
client/src/app/menu/components/CategoryCard.tsx
client/src/app/menu/components/MenuSections.tsx
client/src/app/menu/components/NoResults.tsx
client/src/app/menu/components/ProductCard.tsx
client/src/app/menu/components/ProductDetails.tsx
client/src/app/menu/components/PromotionsBanner.tsx
client/src/app/menu/components/SearchBar.tsx
client/src/app/menu/components/SpecialOffers.tsx
client/src/app/menu/layout.tsx
client/src/app/menu/loading.tsx
client/src/app/menu/page.tsx
client/src/app/orders/[State]/page.tsx
client/src/app/orders/components/OrderItem.tsx
client/src/app/orders/layout.tsx
client/src/app/orders/page.tsx
client/src/app/page.tsx
client/src/app/photo/booking/page.tsx
client/src/app/photo/page.tsx
client/src/app/photo/placeholder-images.json
client/src/app/photo/placeholderImages.ts
client/src/app/privacy/components/PrivacyContent.tsx
client/src/app/privacy/loading.tsx
client/src/app/privacy/page.tsx
client/src/app/profile/components/PersonalInformation.tsx
client/src/app/profile/edit/page.tsx
client/src/app/profile/loading.tsx
client/src/app/profile/page.tsx
client/src/app/terms/components/TermsContent.tsx
client/src/app/terms/loading.tsx
client/src/app/terms/page.tsx
client/src/components/components.css
client/src/components/Navbar/AuthButton.tsx
client/src/components/Navbar/DesktopNavbar.tsx
client/src/components/Navbar/MobileMenu.tsx
client/src/components/Navbar/MobileNavbar.tsx
client/src/components/Navbar/Navbar.tsx
client/src/components/Navbar/NavbarClient.tsx
client/src/components/Navbar/ProfileMenu.tsx
client/src/components/Navbar/useMenubarToggle.ts
client/src/components/ui/AddOnSelector.tsx
client/src/components/ui/Alert.tsx
client/src/components/ui/Avatar.tsx
client/src/components/ui/Button.tsx
client/src/components/ui/FormField.tsx
client/src/components/ui/Icon.tsx
client/src/components/ui/ImageUpload.tsx
client/src/components/ui/layout/FormField.tsx

client/src/components/ui/layout/Main.tsx
client/src/components/ui/layout/Section.tsx
client/src/components/ui/Link.tsx
client/src/components/ui/Loading.tsx
client/src/components/ui/TextInput.tsx
client/src/components/ui/TimeSlotPicker.tsx
client/src/lib/actions/auth.ts
client/src/lib/actions/profile.ts
client/src/lib/context/CartContext.tsx
client/src/lib/forms/CartProductForm.tsx
client/src/lib/forms/CommentsForm.tsx
client/src/lib/forms/ContactForm.tsx
client/src/lib/forms/EditProfileForm.tsx
client/src/lib/forms/LoginForm.tsx
client/src/lib/forms/RegisterForm.tsx
client/src/lib/forms/UserAddressForm.tsx
client/src/lib/hooks/Cookies/getdelete.ts
client/src/lib/hooks/Cookies/setCookie.ts
client/src/lib/hooks/Cookies/setdelete.ts
client/src/lib/supabase/auth/logout.ts
client/src/lib/supabase/auth/useAuth.ts
client/src/lib/supabase/client.ts
client/src/lib/supabase/comments.ts
client/src/lib/supabase/favorites.ts
client/src/lib/supabase/likes.ts
client/src/lib/supabase/orders.ts
client/src/lib/supabase/orders/orders.ts
client/src/lib/supabase/products/products.ts
client/src/lib/supabase/server.ts
client/src/lib/supabase/storage.ts
client/src/lib/types/CarWashTypes.d.ts
client/src/lib/types/database.types.ts
client/src/lib/types/index.ts
client/src/lib/types/OrderTypes.d.ts
client/src/lib/types/ProductTypes.d.ts
client/src/lib/types/UserTypes.d.ts
client/src/lib/utils.ts
client/src/lib/utils/dateUtils.ts
client/src/proxy.ts
client/supabase_migrations/add_profile_trigger.sql
client/supabase_migrations/create_tables.sql
client/tsconfig.json
</directory_structure>

<files>
This section contains the contents of the repository's files.

<file path="admin/.gitignore">
See https://help.github.com/articles/ignoring-files/ for more about ignoring files.

dependencies
/node_modules
/.pnp
.pnp.*
.yarn/*
!.yarn/patches
!.yarn/plugins
!.yarn/releases
!.yarn/versions

testing
/coverage

```
# next.js
/.next/
/out/

# production
/build

# misc
.DS_Store
*.pem

# debug
npm-debug.log*
yarn-debug.log*
yarn-error.log*
.pnpm-debug.log*

# env files (can opt-in for committing if needed)
.env*

# vercel
.vercel

# typescript
*.tsbuildinfo
next-env.d.ts
</file>

<file path="admin/biome.json">
{
  "$schema": "https://biomejs.dev/schemas/2.2.0/schema.json",
  "vcs": {
    "enabled": true,
    "clientKind": "git",
    "useIgnoreFile": true
  },
  "files": {
    "ignoreUnknown": true,
    "includes": ["**", "!node_modules", "!.next", "!dist", "!build"]
  },
  "formatter": {
    "enabled": true,
    "indentStyle": "space",
    "indentWidth": 2
  },
  "linter": {
    "enabled": true,
    "rules": {
      "recommended": true,
      "suspicious": {
        "noUnknownAtRules": "off"
      }
    },
    "domains": {
      "next": "recommended",
      "react": "recommended"
    }
  },
  "assist": {
    "actions": {
      "source": {
        "organizeImports": "on"
      }
    }
```

```
  }
}
</file>

<file path="admin/package.json">
{
  "name": "admin",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "biome check",
    "format": "biome format --write"
  },
  "dependencies": {
    "@supabase/ssr": "^0.7.0",
    "@supabase/supabase-js": "^2.81.1",
    "date-fns": "^4.1.0",
    "next": "16.0.3",
    "react": "19.2.0",
    "react-dom": "19.2.0",
    "react-icons": "^5.5.0"
  },
  "devDependencies": {
    "@biomejs/biome": "2.2.0",
    "@tailwindcss/postcss": "^4",
    "@types/node": "^20",
    "@types/react": "^19",
    "@types/react-dom": "^19",
    "tailwindcss": "^4",
    "typescript": "^5"
  }
}
</file>

<file path="admin/postcss.config.mjs">
const config = {
  plugins: {
    "@tailwindcss/postcss": {},
  },
};

export default config;
</file>

<file path="admin/proxy.ts">
import { createServerClient } from "@supabase/ssr";
import { type NextRequest, NextResponse } from "next/server";
import { getSupabaseConfig } from "@/lib/supabase/config";

const AUTH_ROUTES = ["/login", "/register"];

export async function proxy(req: NextRequest) {
  const res = NextResponse.next({
    request: {
      headers: req.headers,
    },
  });

  const { url, anonKey } = getSupabaseConfig();
  const supabase = createServerClient(url, anonKey, {
    cookies: {
```

```
      getAll() {
        return req.cookies.getAll();
      },
      setAll(cookiesToSet) {
        cookiesToSet.forEach(({ name, value, options }) => {
          res.cookies.set(name, value, options);
        });
      },
    },
  });

  const {
    data: { user },
  } = await supabase.auth.getUser();

  const pathname = req.nextUrl.pathname;
  const isAuthRoute = AUTH_ROUTES.some((route) => pathname.startsWith(route));

  if (!user && !isAuthRoute) {
    const redirectUrl = req.nextUrl.clone();
    redirectUrl.pathname = "/login";
    redirectUrl.searchParams.set("redirectTo", pathname);
    return NextResponse.redirect(redirectUrl);
  }

  if (user && isAuthRoute) {
    return NextResponse.redirect(new URL("/dashboard", req.url));
  }

  return res;
}

export const config = {
  matcher: ["/((?!_next/static|_next/image|favicon.ico).*)"],
};
</file>

<file path="admin/public/file.svg">
<svg fill="none" viewBox="0 0 16 16" xmlns="http://www.w3.org/2000/svg"><path
d="M14.5 13.5V5.41a1 1 0 0 0-.3-.7L9.8.29A1 1 0 0 0 9.08 0H1.5v13.5A2.5 2.5 0 0
0 4 16h8a2.5 2.5 0 0 0 2.5-2.5m-1.5 0v-7H8v-5H3v12a1 1 0 0 0 1 1h8a1 1 0 0 0 1-
1M9.5 5V2.12L12.38 5zM5.13 5h-.62v1.25h2.12V5zm-.62 3h7.12v1.25H4.5zm.62
3h-.62v1.25h7.12V11z" clip-rule="evenodd" fill="#666"
fill-rule="evenodd"/></svg>
</file>

<file path="admin/public/globe.svg">
<svg fill="none" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 16 16"><g clip-
path="url(#a)"><path fill-rule="evenodd" clip-rule="evenodd" d="M10.27 14.1a6.5
6.5 0 0 0 3.67-3.45q-1.24.21-2.7.34-.31 1.83-.97 3.1M8 16A8 8 0 1 0 8 0a8 8 0 0
0 0 16m.48-1.52a7 7 0 0 1-.96 0H7.5a4 4 0 0 1-.84-1.32q-.38-.89-.63-2.08a40 40 0
0 0 3.92 0q-.25 1.2-.63 2.08a4 4 0 0 1-.84 1.31zm2.94-4.76q1.66-.15 2.95-.43a7 7
0 0 0 0-2.58q-1.3-.27-2.95-.43a18 18 0 0 1 0 3.44m-1.27-3.54a17 17 0 0 1 0 3.64
39 39 0 0 1-4.3 0 17 17 0 0 1 0-3.64 39 39 0 0 1 4.3 0m1.1-1.17q1.45.13
2.69.34a6.5 6.5 0 0 0-3.67-3.44q.65 1.26.98 3.1M8.48 1.5l.01.02q.41.37.84
1.31.38.89.63 2.08a40 40 0 0 0-3.92 0q.25-1.2.63-2.08a4 4 0 0 1 .85-1.32 7 7 0 0
1 .96 0m-2.75.4a6.5 6.5 0 0 0-3.67 3.44 29 29 0 0 1 2.7-.34q.31-1.83.97-3.1M4.58
6.28q-1.66.16-2.95.43a7 7 0 0 0 0 2.58q1.3.27 2.95.43a18 18 0 0 1 0-3.44m.17
4.71q-1.45-.12-2.69-.34a6.5 6.5 0 0 0 3.67 3.44q-.65-1.27-.98-3.1"
fill="#666"/></g><defs><clipPath id="a"><path fill="#fff" d="M0
0h16v16H0z"/></clipPath></defs></svg>
</file>

<file path="admin/public/next.svg">
```

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 394 80"><path fill="#000" d="M262 0h68.5v12.7h-27.2v66.6h-13.6V12.7H262V0ZM149 0v12.7H94v20.4h44.3v12.6H94v21h55v12.6H80.5V0h68.7zm34.3 0h-17.8l63.8 79.4h17.9l-32-39.7 32-39.6h-17.9l-23 28.6-23-28.6zm18.3 56.7-9-11-27.1 33.7h17.8l18.3-22.7z"/><path fill="#000" d="M81 79.3 17 0H0v79.3h13.6V17l50.2 62.3H81Zm252.6-.4c-1 0-1.8-.4-2.5-1s-1.1-1.6-1.1-2.6.3-1.8 1-2.5 1.6-1 2.6-1 1.8.3 2.5 1a3.4 3.4 0 0 1 .6 4.3 3.7 3.7 0 0 1-3 1.8zm23.2-33.5h6v23.3c0 2.1-.4 4-1.3 5.5a9.1 9.1 0 0 1-3.8 3.5c-1.6.8-3.5 1.3-5.7 1.3-2 0-3.7-.4-5.3-1s-2.8-1.8-3.7-3.2c-.9-1.3-1.4-3-1.4-5h6c.1.8.3 1.6.7 2.2s1 1.2 1.6 1.5c.7.4 1.5.5 2.4.5 1 0 1.8-.2 2.4-.6a4 4 0 0 0 1.6-1.8c.3-.8.5-1.8.5-3V45.5zm30.9 9.1a4.4 4.4 0 0 0-2-3.3 7.5 7.5 0 0 0-4.3-1.1c-1.3 0-2.4.2-3.3.5-.9.4-1.6 1-2 1.6a3.5 3.5 0 0 0-.3 4c.3.5.7.9 1.3 1.2l1.8 1 2 .5 3.2.8c1.3.3 2.5.7 3.7 1.2a13 13 0 0 1 3.2 1.8 8.1 8.1 0 0 1 3 6.5c0 2-.5 3.7-1.5 5.1a10 10 0 0 1-4.4 3.5c-1.8.8-4.1 1.2-6.8 1.2-2.6 0-4.9-.4-6.8-1.2-2-.8-3.4-2-4.5-3.5a10 10 0 0 1-1.7-5.6h6a5 5 0 0 0 3.5 4.6c1 .4 2.2.6 3.4.6 1.3 0 2.5-.2 3.5-.6 1-.4 1.8-1 2.4-1.7a4 4 0 0 0 .8-2.4c0-.9-.2-1.6-.7-2.2a11 11 0 0 0-2.1-1.4l-3.2-1-3.8-1c-2.8-.7-5-1.7-6.6-3.2a7.2 7.2 0 0 1-2.4-5.7 8 8 0 0 1 1.7-5 10 10 0 0 1 4.3-3.5c2-.8 4-1.2 6.4-1.2 2.3 0 4.4.4 6.2 1.2 1.8.8 3.2 2 4.3 3.4 1 1.4 1.5 3 1.5 5h-5.8z"/></svg>
</file>

<file path="admin/public/vercel.svg">
<svg fill="none" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 1155 1000"><path d="m577.3 0 577.4 1000H0z" fill="#fff"/></svg>
</file>

<file path="admin/public/window.svg">
<svg fill="none" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 16 16"><path fill-rule="evenodd" clip-rule="evenodd" d="M1.5 2.5h13v10a1 1 0 0 1-1 1h-11a1 1 0 0 1-1-1zM0 1h16v11.5a2.5 2.5 0 0 1-2.5 2.5h-11A2.5 2.5 0 0 1 0 12.5zm3.75 4.5a.75.75 0 1 0 0-1.5.75.75 0 0 0 0 1.5M7 4.75a.75.75 0 1 1-1.5 0 .75.75 0 0 1 1.5 0m1.75.75a.75.75 0 1 0 0-1.5.75.75 0 0 0 0 1.5" fill="#666"/></svg>
</file>

<file path="admin/README.md">
This is the administrative portal for PA Catering, built with the Next.js App Router, Supabase, and Tailwind CSS.

Prerequisites

1. Create (or reuse) a Supabase project.
2. Copy the public URL and anon key into an `.env.local` file:

```bash
NEXT_PUBLIC_SUPABASE_URL=...
NEXT_PUBLIC_SUPABASE_PUBLISHABLE_KEY=...
# Optional: require invite code for new admins
ADMIN_INVITE_CODE=super-secure-code
```

Available scripts

```bash
npm run dev     # Start the local dev server
npm run build   # Create a production build
npm run start   # Run the production build locally
npm run lint    # Biome lint pass
npm run format  # Biome format with write
```

Directory overview

- `src/app`: App Router routes (auth, dashboard, menu, orders, etc.).
- `src/lib/supabase`: Shared browser/server Supabase clients.
- `src/lib/types`: Product, order, and user domain models used across features.

```
## Deployment

Deploy like any other Next.js project (Vercel, Render, Docker, etc.). Ensure the
Supabase environment variables are provided in the hosting platform.
</file>

<file path="admin/src/app/(auth)/login/page.tsx">
import Link from "next/link";
import { AuthShell } from "@/components/auth/AuthShell";
import { LoginForm } from "@/components/auth/LoginForm";

type SearchParams = {
  redirectTo?: string;
  registered?: string;
  error?: string;
};

export default function LoginPage({
  searchParams,
}: {
  searchParams: SearchParams;
}) {
  const redirectTo = searchParams?.redirectTo;
  const registered = searchParams?.registered === "1";
  const error = searchParams?.error;

  const initialMessage = registered
    ? "Account created successfully. You can sign in now."
    : undefined;
  const description =
    error === "not_authorized"
      ? "Your account lacks admin permissions. Contact Operations for access."
      : "Access internal dashboards to manage menus, orders, and logistics.";

  return (
    <AuthShell
      title="Admin sign in"
      description={description}
      footer={
        <p>
          Need access?{" "}
          <Link
            href="/register"
            className="text-indigo-300 underline-offset-4 hover:underline"
          >
            Request an admin account
          </Link>
        </p>
      }
    >
      <LoginForm redirectTo={redirectTo} initialMessage={initialMessage} />
    </AuthShell>
  );
}
</file>

<file path="admin/src/app/(auth)/register/page.tsx">
import Link from "next/link";
import { AuthShell } from "@/components/auth/AuthShell";
import { RegisterForm } from "@/components/auth/RegisterForm";

type SearchParams = {
  error?: string;
```

```
};

export default function RegisterPage({
  searchParams,
}: {
  searchParams: SearchParams;
}) {
  return (
    <AuthShell
      title="Invite-only registration"
      description="Only trusted operations staff should create new admin
accounts."
      footer={
        <p>
          Already onboarded?{" "}
          <Link
            href="/login"
            className="text-indigo-300 underline-offset-4 hover:underline"
          >
            Sign in
          </Link>
        </p>
      }
    >
      <RegisterForm initialError={searchParams?.error} />
    </AuthShell>
  );
}
</file>

<file path="admin/src/app/(dashboard)/contact/page.tsx">
import { DashboardShell } from "@/components/layout/DashboardShell";
import { ContactBoard } from "@/components/contact/ContactBoard";
import { fetchContactSubmissions } from "@/lib/data/contact";

export default async function ContactPage() {
    const contacts = await fetchContactSubmissions();

    return (
        <DashboardShell
            title="Contact Submissions"
            description="View all contact form submissions from customers."
        >
            <ContactBoard contacts={contacts} />
        </DashboardShell>
    );
}
</file>

<file path="admin/src/app/(dashboard)/dashboard/page.tsx">
import { LowInventoryCard } from "@/components/dashboard/LowInventoryCard";
import { RecentOrdersCard } from "@/components/dashboard/RecentOrdersCard";
import { RevenueTrendCard } from "@/components/dashboard/RevenueTrendCard";
import { StatsGrid } from "@/components/dashboard/StatsGrid";
import { DashboardShell } from "@/components/layout/DashboardShell";
import { fetchDashboardData } from "@/lib/data/dashboard";

export default async function DashboardPage() {
  const data = await fetchDashboardData();

  return (
    <div className="space-y-6">
      <DashboardShell
        title="Executive summary"
```

```
        description="Live snapshot of menu demand, fulfillment, and stock
health."
      >
        <StatsGrid stats={data.stats} />
      </DashboardShell>

      <div className="grid gap-6 lg:grid-cols-3">
        <div className="lg:col-span-2">
          <RecentOrdersCard orders={data.recentOrders} />
        </div>
        <div className="space-y-6">
          <RevenueTrendCard series={data.revenueSeries} />
          <LowInventoryCard items={data.lowInventory} />
        </div>
      </div>
    </div>
  );
}
</file>

<file path="admin/src/app/(dashboard)/featured-items/page.tsx">
import { DashboardShell } from "@/components/layout/DashboardShell";
import { FeaturedItemsBoard } from
"@/components/featured-items/FeaturedItemsBoard";
import { fetchFeaturedItems } from "@/lib/data/testimonials";

export default async function FeaturedItemsPage() {
    const items = await fetchFeaturedItems();

    return (
        <DashboardShell
            title="Featured Items"
            description="Manage featured menu items showcased to customers."
        >
            <FeaturedItemsBoard items={items} />
        </DashboardShell>
    );
}
</file>

<file path="admin/src/app/(dashboard)/orders/page.tsx">
import { DashboardShell } from "@/components/layout/DashboardShell";
import { OrdersBoard } from "@/components/orders/OrdersBoard";
import { fetchOrdersBoard } from "@/lib/data/orders";

export default async function OrdersPage() {
  const { orders } = await fetchOrdersBoard();

  return (
    <DashboardShell
      title="Fulfilment queue"
      description="Assign, track, and complete customer orders in real time."
    >
      <OrdersBoard orders={orders} />
    </DashboardShell>
  );
}
</file>

<file path="admin/src/app/(dashboard)/photo-bookings/page.tsx">
import { DashboardShell } from "@/components/layout/DashboardShell";
import { PhotoBookingsBoard } from
"@/components/photo-bookings/PhotoBookingsBoard";
import { fetchPhotoBookings } from "@/lib/data/photo-bookings";
```

```
export default async function PhotoBookingsPage() {
    const bookings = await fetchPhotoBookings();

    return (
        <DashboardShell
            title="360 Photo Booth Bookings"
            description="Manage all photo booth booking requests and
reservations."
        >
            <PhotoBookingsBoard bookings={bookings} />
        </DashboardShell>
    );
}
</file>

<file path="admin/src/app/(dashboard)/testimonials/page.tsx">
import { DashboardShell } from "@/components/layout/DashboardShell";
import { TestimonialsBoard } from "@/components/testimonials/TestimonialsBoard";
import { fetchTestimonials } from "@/lib/data/testimonials";

export default async function TestimonialsPage() {
    const testimonials = await fetchTestimonials();

    return (
        <DashboardShell
            title="Testimonials"
            description="Manage customer testimonials and reviews for your
business."
        >
            <TestimonialsBoard testimonials={testimonials} />
        </DashboardShell>
    );
}
</file>

<file path="admin/src/app/globals.css">
@import "tailwindcss";

:root {
  --background: #ffffff;
  --foreground: #171717;
}

@theme inline {
  --color-background: var(--background);
  --color-foreground: var(--foreground);
  --font-sans: var(--font-geist-sans);
  --font-mono: var(--font-geist-mono);
}

@media (prefers-color-scheme: dark) {
  :root {
    --background: #0a0a0a;
    --foreground: #ededed;
  }
}

body {
  background: var(--background);
  color: var(--foreground);
  font-family: Arial, Helvetica, sans-serif;
}
</file>
```

<file path="admin/src/app/page.tsx">
```
import { redirect } from "next/navigation";

export default function Home() {
  redirect("/dashboard");
}
```
</file>

<file path="admin/src/components/forms/SubmitButton.tsx">
```
"use client";

import { useFormStatus } from "react-dom";

type Props = {
  label: string;
  loadingLabel?: string;
  className?: string;
};

export const SubmitButton = ({
  label,
  loadingLabel = "Working...",
  className,
}: Props) => {
  const { pending } = useFormStatus();

  return (
    <button
      type="submit"
      disabled={pending}
      className={`inline-flex w-full items-center justify-center rounded-lg bg-
indigo-500 px-4 py-2 font-semibold text-white transition hover:bg-indigo-400
disabled:opacity-60 ${className ?? ""}`}
    >
      {pending ? loadingLabel : label}
    </button>
  );
};
```
</file>

<file path="admin/src/components/menu/products/AddProductPanel.tsx">
```
import { DashboardShell } from "@/components/layout/DashboardShell";
import { AddProductForm } from "./AddProductForm";

type Props = {
  categories: string[];
};

export const AddProductPanel = ({ categories }: Props) => (
  <DashboardShell
    title="Add menu item"
    description="Create a new product and sync it directly to Supabase."
  >
    <AddProductForm categories={categories} />
  </DashboardShell>
);
```
</file>

<file path="admin/src/lib/auth/index.ts">
```
// Export actions and types (safe for client components)
export * from "./actions";
export * from "./types";
```
</file>

```
// Session functions are server-only - import directly from "./session" in
server components
</file>

<file path="admin/src/lib/auth/types.ts">
export type AuthActionState = {
  error?: string;
  message?: string;
};
</file>

<file path="admin/src/lib/data/categories.ts">
import { createSupabaseServerClient } from "@/lib/supabase/server";
import type { CategoryRecord } from "@/lib/types/database-records";

export const fetchCategories = async (): Promise<CategoryRecord[]> => {
    const supabase = await createSupabaseServerClient();
    const { data, error } = await supabase
        .from("products_category")
        .select("*")
        .order("category_name", { ascending: true });

    if (error) {
        console.error("Failed to fetch categories", error);
        return [];
    }

    return (data ?? []) as CategoryRecord[];
};
</file>

<file path="admin/src/lib/data/comments.ts">
import { createSupabaseServerClient } from "@/lib/supabase/server";
import type { CommentRecord } from "@/lib/types/database-records";

export const fetchComments = async (): Promise<CommentRecord[]> => {
    const supabase = await createSupabaseServerClient();
    const { data, error } = await supabase
        .from("comments")
        .select("*")
        .order("created_at", { ascending: false });

    if (error) {
        console.error("Failed to fetch comments", error);
        return [];
    }

    return (data ?? []) as CommentRecord[];
};
</file>

<file path="admin/src/lib/data/contact.ts">
import { createSupabaseServerClient } from "@/lib/supabase/server";
import type { ContactRecord } from "@/lib/types/database-records";

export const fetchContactSubmissions = async (): Promise<ContactRecord[]> => {
    const supabase = await createSupabaseServerClient();
    const { data, error } = await supabase
        .from("contact")
        .select("*")
        .order("created_at", { ascending: false });

    if (error) {
        console.error("Failed to fetch contact submissions", error);
```

```
        return [];
    }

    return (data ?? []) as ContactRecord[];
};
</file>

<file path="admin/src/lib/data/favorites.ts">
import { createSupabaseServerClient } from "@/lib/supabase/server";
import type { UserFavoriteRecord } from "@/lib/types/database-records";

export const fetchUserFavorites = async (): Promise<UserFavoriteRecord[]> => {
    const supabase = await createSupabaseServerClient();
    const { data, error } = await supabase
        .from("user_favorites")
        .select("*")
        .order("created_at", { ascending: false });

    if (error) {
        console.error("Failed to fetch user favorites", error);
        return [];
    }

    return (data ?? []) as UserFavoriteRecord[];
};
</file>

<file path="admin/src/lib/data/photo-bookings.ts">
import { createSupabaseServerClient } from "@/lib/supabase/server";
import type { PhotoBookingRecord } from "@/lib/types/database-records";

export const fetchPhotoBookings = async (): Promise<PhotoBookingRecord[]> => {
    const supabase = await createSupabaseServerClient();
    const { data, error } = await supabase
        .from("photo_boot_bookings")
        .select("*")
        .order("created_at", { ascending: false });

    if (error) {
        console.error("Failed to fetch photo bookings", error);
        return [];
    }

    return (data ?? []) as PhotoBookingRecord[];
};
</file>

<file path="admin/src/lib/data/profiles.ts">
import { createSupabaseServerClient } from "@/lib/supabase/server";
import type { ProfileRecord } from "@/lib/types/database-records";

export const fetchProfiles = async (): Promise<ProfileRecord[]> => {
    const supabase = await createSupabaseServerClient();
    const { data, error } = await supabase
        .from("profiles")
        .select("*")
        .order("created_at", { ascending: false });

    if (error) {
        console.error("Failed to fetch profiles", error);
        return [];
    }

    return (data ?? []) as ProfileRecord[];
```

```
};
</file>

<file path="admin/src/lib/data/testimonials.ts">
import { createSupabaseServerClient } from "@/lib/supabase/server";
import type { TestimonialRecord, FeaturedItemRecord } from "@/lib/types";

export const fetchTestimonials = async (): Promise<TestimonialRecord[]> => {
    const supabase = await createSupabaseServerClient();
    const { data, error } = await supabase
        .from("testimonials")
        .select("*")
        .order("created_at", { ascending: false });

    if (error) {
        console.error("Failed to fetch testimonials", error);
        return [];
    }

    return (data ?? []) as TestimonialRecord[];
};

export const fetchFeaturedItems = async (): Promise<FeaturedItemRecord[]> => {
    const supabase = await createSupabaseServerClient();
    const { data, error } = await supabase
        .from("featured_items")
        .select("*")
        .order("created_at", { ascending: false });

    if (error) {
        console.error("Failed to fetch featured items", error);
        return [];
    }

    return (data ?? []) as FeaturedItemRecord[];
};
</file>

<file path="admin/src/lib/supabase/client.ts">
"use client";

import { createBrowserClient } from "@supabase/ssr";
import { getSupabaseConfig } from "./config";

export const createSupabaseBrowserClient = () =>
  createBrowserClient(getSupabaseConfig().url, getSupabaseConfig().anonKey);

export type SupabaseBrowserClient = ReturnType<
  typeof createSupabaseBrowserClient
>;
</file>

<file path="admin/src/lib/supabase/index.ts">
export * from "./client";
export * from "./server";
</file>

<file path="admin/src/lib/types/base.ts">
export interface Reward {
  name: string;
  pointsrequired: number;
}

export interface Yellowemption {
```

```
  name: string;
  date: string;
}

export type PaymentMethod = "instore" | "Cepitec transfer";

// Compatibility aliases that mirror the client naming convention
export type yellowemption = Yellowemption;
export type paymentMethod = PaymentMethod;
</file>

<file path="admin/src/lib/types/testimonials.ts">
export interface TestimonialRecord {
    id: string;
    text: string;
    author: string;
    rating: number;
    likes: string[];
    comments: unknown[];
    created_at: string;
}

export interface FeaturedItemRecord {
    id: string;
    name: string;
    description: string;
    image_url: string | null;
    likes: string[];
    comments: unknown[];
    created_at: string;
}
</file>

<file path="admin/tsconfig.json">
{
  "compilerOptions": {
    "target": "ES2017",
    "lib": ["dom", "dom.iterable", "esnext"],
    "allowJs": true,
    "skipLibCheck": true,
    "strict": true,
    "noEmit": true,
    "esModuleInterop": true,
    "module": "esnext",
    "moduleResolution": "bundler",
    "resolveJsonModule": true,
    "isolatedModules": true,
    "jsx": "react-jsx",
    "incremental": true,
    "plugins": [
      {
        "name": "next"
      }
    ],
    "paths": {
      "@/*": ["./src/*"]
    }
  },
  "include": [
    "next-env.d.ts",
    "**/*.ts",
    "**/*.tsx",
    ".next/types/**/*.ts",
    ".next/dev/types/**/*.ts",
```

```
    "**/*.mts"
  ],
  "exclude": ["node_modules"]
}
</file>

<file path="client/src/app/(auth)/login/loading.tsx">
import Loading from '@/components/ui/Loading';

export default function LoadingPage() {
      return <Loading message='Loading Login Page...' />;
}
</file>

<file path="client/src/app/(auth)/login/page.tsx">
import Main from "@/components/ui/layout/Main";
import LoginForm from "@/lib/forms/LoginForm";


export default function LoginPage() {


      return (
            <Main tittle="Log In">
                  <LoginForm />
            </Main>
      );
}
</file>

<file path="client/src/app/(auth)/register/loading.tsx">
import Loading from '@/components/ui/Loading';

export default function LoadingPage() {
      return <Loading message='Loading Registration Page...' />;
}
</file>

<file path="client/src/app/(auth)/register/page.tsx">
import Main from '@/components/ui/layout/Main';
import RegisterForm from '@/lib/forms/RegisterForm';

const RegisterPage = () => {

      return (
            <Main tittle='Register'>
                  <RegisterForm />
            </Main>
      );
};

export default RegisterPage;
</file>

<file path="client/src/app/api/seed/route.ts">
import { NextResponse } from 'next/server';
import { createClient } from '@/lib/supabase/server';
import { readFileSync } from 'fs';
import { join } from 'path';

export async function POST() {
  try {
    const supabase = await createClient();
```

```
    // Check if user is authenticated (optional - you can remove this if you
want public seeding)
    const { data: { user }, error: authError } = await supabase.auth.getUser();

    // Optional: Only allow seeding if user is authenticated
    // if (authError || !user) {
    //   return NextResponse.json({ error: 'Unauthorized' }, { status: 401 });
    // }

    // Read the SQL file
    const sqlPath = join(process.cwd(), 'supabase_migrations',
'seed_products.sql');
    const sqlContent = readFileSync(sqlPath, 'utf-8');

    // Execute the SQL
    // Note: Supabase client doesn't have a direct execute method for raw SQL
    // We'll need to use the REST API or RPC function, or parse and insert row
by row
    // For now, let's parse the INSERT statements and execute them

    const insertStatements = sqlContent
      .split(';')
      .map(s => s.trim())
      .filter(s => s.length > 0 && s.toUpperCase().startsWith('INSERT'));

    if (insertStatements.length === 0) {
      return NextResponse.json({ error: 'No INSERT statements found' },
{ status: 400 });
    }

    // Parse the first INSERT statement to extract values
    const mainInsert = insertStatements[0];
    const valuesMatch = mainInsert.match(/VALUES\s*([\s\S]+)/i);

    if (!valuesMatch) {
      return NextResponse.json({ error: 'Could not parse INSERT statement' },
{ status: 400 });
    }

    // Parse values - this is a simplified parser
    const valuesString = valuesMatch[1];
    const rows: Array<{
      name: string;
      category: string;
      description: string | null;
      price: number;
      image_url: string | null;
      badge: string | null;
      stock: number;
    }> = [];

    // Split by rows (each row is in parentheses)
    const rowMatches = valuesString.matchAll(/\(([^)]+)\)/g);

    for (const match of rowMatches) {
      const rowValues = match[1].split(',').map(v => v.trim());

      if (rowValues.length >= 7) {
        const name = rowValues[0].replace(/^'|'$/g, '');
        const category = rowValues[1].replace(/^'|'$/g, '');
        const description = rowValues[2] === 'NULL' ? null :
rowValues[2].replace(/^'|'$/g, '');
        const price = parseFloat(rowValues[3]);
        const image_url = rowValues[4] === 'NULL' ? null :
```

```
rowValues[4].replace(/^'|'$/g, '');
        const badge = rowValues[5] === 'NULL' ? null :
rowValues[5].replace(/^'|'$/g, '');
        const stock = parseInt(rowValues[6], 10);

        rows.push({
          name,
          category,
          description,
          price,
          image_url,
          badge,
          stock,
        });
      }
    }

    // Insert products in batches
    const batchSize = 10;
    let inserted = 0;
    let errors: string[] = [];

    for (let i = 0; i < rows.length; i += batchSize) {
      const batch = rows.slice(i, i + batchSize);

      // @ts-ignore - Supabase type inference issue with Database types
      const { error } = await supabase.from('products').insert(batch);

      if (error) {
        errors.push(`Batch ${Math.floor(i / batchSize) + 1}: ${error.message}`);
      } else {
        inserted += batch.length;
      }
    }

    if (errors.length > 0) {
      return NextResponse.json(
        {
          success: false,
          message: `Inserted ${inserted} products, but encountered errors`,
          errors
        },
        { status: 207 } // Multi-Status
      );
    }

    return NextResponse.json(
      {
        success: true,
        message: `Successfully seeded ${inserted} products`
      },
      { status: 200 }
    );
  } catch (error) {
    console.error('Seed error:', error);
    return NextResponse.json(
      {
        success: false,
        error: error instanceof Error ? error.message : 'Unknown error occurred'
      },
      { status: 500 }
    );
  }
}
```

</file>

<file path="client/src/components/ui/ImageUpload.tsx">
```
'use client';

import { useState, useRef } from 'react';
import { FaUpload, FaTimes } from 'react-icons/fa';
import Image from 'next/image';
import { uploadImage } from '@/lib/supabase/storage';

type Props = {
    defaultValue?: string | null;
    onChange: (url: string) => void;
    folder?: string;
    label?: string;
};

export const ImageUpload = ({
    defaultValue,
    onChange,
    folder = 'uploads',
    label = 'Upload Image',
}: Props) => {
    const [preview, setPreview] = useState<string | null>(defaultValue || null);
    const [uploading, setUploading] = useState(false);
    const [value, setValue] = useState<string>(defaultValue || '');
    const fileInputRef = useRef<HTMLInputElement>(null);

    const handleFileChange = async (e: React.ChangeEvent<HTMLInputElement>) => {
        const file = e.target.files?.[0];
        if (!file) return;

        // Create local preview
        const objectUrl = URL.createObjectURL(file);
        setPreview(objectUrl);
        setUploading(true);

        try {
            const publicUrl = await uploadImage(file, folder);
            setValue(publicUrl);
            onChange(publicUrl);
        } catch (error) {
            console.error('Upload failed:', error);
            alert('Failed to upload image. Please try again.');
            setPreview(defaultValue || null);
            setValue(defaultValue || '');
        } finally {
            setUploading(false);
        }
    };

    const handleRemove = () => {
        setPreview(null);
        setValue('');
        onChange('');
        if (fileInputRef.current) {
            fileInputRef.current.value = '';
        }
    };

    return (
        <div className='space-y-4'>
            <div className='flex items-center gap-4'>
                {preview ? (
```

```
                    <div className='relative h-40 w-40 overflow-hidden rounded-
lg border-2 border-yellow-500/30 bg-black/30'>
                        <Image
                            src={preview}
                            alt='Preview'
                            fill
                            className='object-cover'
                        />
                        <button
                            type='button'
                            onClick={handleRemove}
                            className='absolute right-2 top-2 rounded-full bg-
black/70 p-1.5 text-white hover:bg-red-600 transition-colors'
                        >
                            <FaTimes className='h-4 w-4' />
                        </button>
                        {uploading && (
                            <div className='absolute inset-0 flex items-center
justify-center bg-black/60'>
                                <div className='h-8 w-8 animate-spin rounded-
full border-4 border-yellow-500 border-t-transparent' />
                            </div>
                        )}
                    </div>
                ) : (
                    <button
                        type='button'
                        onClick={() => fileInputRef.current?.click()}
                        className='flex h-40 w-40 flex-col items-center justify-
center gap-2 rounded-lg border-2 border-dashed border-yellow-500/40 bg-black/20
hover:bg-black/40 hover:border-yellow-500/60 transition-all'
                    >
                        <FaUpload className='h-8 w-8 text-yellow-500' />
                        <span className='text-sm text-yellow-500 font-
medium'>{label}</span>
                    </button>
                )}
                <input
                    ref={fileInputRef}
                    type='file'
                    accept='image/*'
                    onChange={handleFileChange}
                    className='hidden'
                />
            </div>
            <input type='hidden' name='photoURL' value={value} />
        </div>
    );
};
</file>

<file path="client/src/lib/actions/profile.ts">
"use server";

import { createClient } from "@/lib/supabase/server";

export type ProfileActionState = {
  success: boolean;
  error?: string;
};

export async function updateProfileAction(
  _prevState: ProfileActionState | null,
  formData: FormData
```

```
): Promise<ProfileActionState> {
  const displayName = String(formData.get("displayName") ?? "").trim();
  const phoneNumber = String(formData.get("phoneNumber") ?? "").trim() || null;
  const photoURL = String(formData.get("photoURL") ?? "").trim() || null;

  if (!displayName) {
    return {
      success: false,
      error: "Display name is required.",
    };
  }

  try {
    const supabase = await createClient();
    const {
      data: { user },
      error: authError,
    } = await supabase.auth.getUser();

    if (authError || !user) {
      return {
        success: false,
        error: "User not authenticated.",
      };
    }

    // Update user metadata
    const { error: updateError } = await supabase.auth.updateUser({
      data: {
        display_name: displayName,
        phone_number: phoneNumber,
        photoURL: photoURL,
      },
    });

    if (updateError) {
      return {
        success: false,
        error: updateError.message,
      };
    }

    // Also update the profiles table
    const { error: profileError } = await supabase
      .from("profiles")
      // @ts-ignore - Supabase type inference issue with Database types
      .update({
        display_name: displayName,
        phone: phoneNumber,
      })
      .eq("id", user.id);

    if (profileError) {
      console.error("Error updating profile:", profileError);
      // Don't fail if profile update fails, but log it
    }

    return {
      success: true,
    };
  } catch (error) {
    return {
      success: false,
      error: error instanceof Error ? error.message : "An unexpected error
```

```
occurred while updating profile.",
    };
  }
}

export async function updateAddressAction(
  _prevState: ProfileActionState | null,
  formData: FormData
): Promise<ProfileActionState> {
  const address = String(formData.get("address") ?? "").trim();
  const city = String(formData.get("city") ?? "").trim();
  const state = String(formData.get("state") ?? "").trim();
  const zipCode = String(formData.get("zipCode") ?? "").trim();
  const country = String(formData.get("country") ?? "").trim();

  if (!address || !city || !state || !zipCode || !country) {
    return {
      success: false,
      error: "All address fields are required.",
    };
  }

  try {
    const supabase = await createClient();
    const {
      data: { user },
      error: authError,
    } = await supabase.auth.getUser();

    if (authError || !user) {
      return {
        success: false,
        error: "User not authenticated.",
      };
    }

    // Update user metadata with address information
    const { error: updateError } = await supabase.auth.updateUser({
      data: {
        address: address,
        city: city,
        state: state,
        zipCode: zipCode,
        country: country,
      },
    });

    if (updateError) {
      return {
        success: false,
        error: updateError.message,
      };
    }

    return {
      success: true,
    };
  } catch (error) {
    return {
      success: false,
      error: error instanceof Error ? error.message : "An unexpected error
occurred while updating address.",
    };
  }
```

```
}
</file>

<file path="client/src/lib/supabase/storage.ts">
import { createClient } from "./client";

export const STORAGE_BUCKET = "pa-luxe-creation";

export const uploadImage = async (file: File, path: string) => {
    const supabase = createClient();
    const fileExt = file.name.split(".").pop();
    const fileName = `${Math.random().toString(36).substring(2)}.${fileExt}`;
    const filePath = `${path}/${fileName}`;

    const { error: uploadError } = await supabase.storage
        .from(STORAGE_BUCKET)
        .upload(filePath, file);

    if (uploadError) {
        throw uploadError;
    }

    const { data } =
supabase.storage.from(STORAGE_BUCKET).getPublicUrl(filePath);

    return data.publicUrl;
};
</file>

<file path="client/src/lib/types/database.types.ts">
export type Json = string | number | boolean | null | { [key: string]: Json |
undefined } | Json[]

export type Database = {
  public: {
    Tables: {
      profiles: {
        Row: {
          id: string
          email: string | null
          display_name: string | null
          phone: string | null
          metadata: Json | null
          created_at: string
          role: string | null
          uid: string | null
          email_verified: boolean | null
          photo_url: string | null
          address: string | null
          city: string | null
          state: string | null
          zip_code: string | null
          country: string | null
          theme: string | null
          tier_status: string | null
          referral_code: string | null
          preferences: Json | null
          saved_payment_methods: Json | null
          updated_at: string | null
          last_login: string | null
        }
        Insert: {
          id?: string
          email?: string | null
```

```
      display_name?: string | null
      phone?: string | null
      metadata?: Json | null
      created_at?: string
      role?: string | null
      uid?: string | null
      email_verified?: boolean | null
      photo_url?: string | null
      address?: string | null
      city?: string | null
      state?: string | null
      zip_code?: string | null
      country?: string | null
      theme?: string | null
      tier_status?: string | null
      referral_code?: string | null
      preferences?: Json | null
      saved_payment_methods?: Json | null
      updated_at?: string | null
      last_login?: string | null
    }
    Update: {
      id?: string
      email?: string | null
      display_name?: string | null
      phone?: string | null
      metadata?: Json | null
      created_at?: string
      role?: string | null
      uid?: string | null
      email_verified?: boolean | null
      photo_url?: string | null
      address?: string | null
      city?: string | null
      state?: string | null
      zip_code?: string | null
      country?: string | null
      theme?: string | null
      tier_status?: string | null
      referral_code?: string | null
      preferences?: Json | null
      saved_payment_methods?: Json | null
      updated_at?: string | null
      last_login?: string | null
    }
    Relationships: []
  }
  products: {
    Row: {
      id: string
      name: string
      slug: string | null
      description: string | null
      price: number | null
      category_name: string | null
      image_url: string | null
      stock: number | null
      likes: string[] | null
      badge: string | null
      created_at: string
      is_hidden: boolean | null
    }
    Insert: {
      id?: string
```

```
      name: string
      slug?: string | null
      description?: string | null
      price?: number | null
      category_name?: string | null
      image_url?: string | null
      stock?: number | null
      likes?: string[] | null
      badge?: string | null
      created_at?: string
      is_hidden?: boolean | null
    }
    Update: {
      id?: string
      name?: string
      slug?: string | null
      description?: string | null
      price?: number | null
      category_name?: string | null
      image_url?: string | null
      stock?: number | null
      likes?: string[] | null
      badge?: string | null
      created_at?: string
      is_hidden?: boolean | null
    }
  }
  products_category: {
    Row: {
      id: string
      category_name: string
      image: string | null
      description: string | null
      created_at: string
      is_hidden: boolean | null
    }
    Insert: {
      id?: string
      category_name: string
      image?: string | null
      description?: string | null
      created_at?: string
      is_hidden?: boolean | null
    }
    Update: {
      id?: string
      category_name?: string
      image?: string | null
      description?: string | null
      created_at?: string
      is_hidden?: boolean | null
    }
  }
  comments: {
    Row: {
      id: string
      product_id: string | null
      user_id: string | null
      user_name: string | null
      body: string
      created_at: string
    }
    Insert: {
      id?: string
```

```
      product_id?: string | null
      user_id?: string | null
      user_name?: string | null
      body: string
      created_at?: string
    }
    Update: {
      id?: string
      product_id?: string | null
      user_id?: string | null
      user_name?: string | null
      body?: string
      created_at?: string
    }
  }
  testimonials: {
    Row: {
      id: string
      text: string
      author: string
      rating: number
      likes: string[] | null
      comments: Json | null
      created_at: string
    }
    Insert: {
      id?: string
      text: string
      author: string
      rating?: number
      likes?: string[] | null
      comments?: Json | null
      created_at?: string
    }
    Update: {
      id?: string
      text?: string
      author?: string
      rating?: number
      likes?: string[] | null
      comments?: Json | null
      created_at?: string
    }
  }
  featured_items: {
    Row: {
      id: string
      name: string
      description: string
      image_url: string | null
      likes: string[] | null
      comments: Json | null
      created_at: string
    }
    Insert: {
      id?: string
      name: string
      description: string
      image_url?: string | null
      likes?: string[] | null
      comments?: Json | null
      created_at?: string
    }
    Update: {
```

```
      id?: string
      name?: string
      description?: string
      image_url?: string | null
      likes?: string[] | null
      comments?: Json | null
      created_at?: string
    }
  }
  contact: {
    Row: {
      id: string
      name: string
      email: string
      phone: string | null
      message: string
      created_at: string
    }
    Insert: {
      id?: string
      name: string
      email: string
      phone?: string | null
      message: string
      created_at?: string
    }
    Update: {
      id?: string
      name?: string
      email?: string
      phone?: string | null
      message?: string
      created_at?: string
    }
  }
  photo_boot_bookings: {
    Row: {
      id: string
      name: string
      email: string
      phone: string | null
      date: string
      time: string
      package: string
      people: number
      message: string | null
      created_at: string
    }
    Insert: {
      id?: string
      name: string
      email: string
      phone?: string | null
      date: string
      time: string
      package: string
      people: number
      message?: string | null
      created_at?: string
    }
    Update: {
      id?: string
      name?: string
      email?: string
```

```
      phone?: string | null
      date?: string
      time?: string
      package?: string
      people?: number
      message?: string | null
      created_at?: string
    }
  }
  orders: {
    Row: {
      id: string
      user_id: string | null
      items: Json
      total_price: number
      total_quantity: number
      status: string | null
      created_at: string
      updated_at: string | null
    }
    Insert: {
      id?: string
      user_id?: string | null
      items: Json
      total_price: number
      total_quantity: number
      status?: string | null
      created_at?: string
      updated_at?: string | null
    }
    Update: {
      id?: string
      user_id?: string | null
      items?: Json
      total_price?: number
      total_quantity?: number
      status?: string | null
      created_at?: string
      updated_at?: string | null
    }
  }
  user_favorites: {
    Row: {
      user_id: string
      product_id: string
      created_at: string | null
    }
    Insert: {
      user_id: string
      product_id: string
      created_at?: string | null
    }
    Update: {
      user_id?: string
      product_id?: string
      created_at?: string | null
    }
  }
}
Views: {
  [_ in never]: never
}
Functions: {
  [_ in never]: never
```

```
    }
    Enums: {
      [_ in never]: never
    }
    CompositeTypes: {
      [_ in never]: never
    }
  }
}
</file>

<file path="admin/next.config.ts">
import type { NextConfig } from "next";

const nextConfig: NextConfig = {
  images: {
    remotePatterns: [
      {
        protocol: "https",
        hostname: "qiksfxnjdfpncpsmzvko.supabase.co",
      },
    ],
  },
};

export default nextConfig;
</file>

<file path="admin/src/app/(auth)/layout.tsx">
import type { ReactNode } from "react";

export default function AuthLayout({ children }: { children: ReactNode }) {
  return (
    <div className="min-h-screen bg-gradient-to-br from-yellow-950 via-
yellow-900 to-yellow-950 px-4 py-10">
      <div className="mx-auto flex max-w-6xl flex-col items-center justify-
center gap-8">
        <div className="text-center text-white">
          <p className="text-sm uppercase tracking-[0.3em] text-indigo-300">
            PA Catering
          </p>
          <h1 className="text-3xl font-semibold">Operations Console</h1>
        </div>
        {children}
      </div>
    </div>
  );
}
</file>

<file path="admin/src/app/(dashboard)/layout.tsx">
import type { ReactNode } from "react";
import { AdminSidebar } from "@/components/layout/AdminSidebar";
import { AdminTopbar } from "@/components/layout/AdminTopbar";
import { requireAdminProfile } from "@/lib/auth/session";

export default async function DashboardLayout({
  children,
}: {
  children: ReactNode;
}) {
  const profile = await requireAdminProfile();

  return (
```

```
    <div className="flex min-h-screen bg-yellow-950 text-white">
      <AdminSidebar />
      <div className="flex flex-1 flex-col">
        <AdminTopbar profile={profile} />
        <main className="flex-1 space-y-6 bg-yellow-950 px-4 py-6 lg:px-8">
          {children}
        </main>
      </div>
    </div>
  );
}
</file>

<file path="admin/src/app/(dashboard)/menu/products/[id]/edit/page.tsx">
import { notFound, redirect } from "next/navigation";
import { DashboardShell } from "@/components/layout/DashboardShell";
import { ProductEditForm } from "@/components/menu/products/ProductEditForm";
import { fetchProductCatalog } from "@/lib/data/products";

type Props = {
    params: Promise<{ id: string }>;
};

export default async function EditProductPage({ params }: Props) {
    const { id } = await params;
    const products = await fetchProductCatalog();
    const product = products.find((p) => p.id === id);

    if (!product) {
        notFound();
    }

    const categories = Array.from(
        new Set(products.map((p) => p.category_name).filter(Boolean)),
    ) as string[];

    return (
        <DashboardShell
            title={`Edit: ${product.name}`}
            description="Update product details, pricing, and availability."
        >
            <div className="mx-auto max-w-2xl">
                <ProductEditForm product={product} categories={categories} />
            </div>
        </DashboardShell>
    );
}
</file>

<file path="admin/src/app/(dashboard)/profiles/page.tsx">
import { DashboardShell } from "@/components/layout/DashboardShell";
import { ProfilesBoard } from "@/components/profiles/ProfilesBoard";
import { fetchProfiles } from "@/lib/data/profiles";
import { fetchProfileOrders, fetchProfileFavorites } from "@/lib/data/profile-
details";

export default async function ProfilesPage() {
    const profiles = await fetchProfiles();

    const handleViewProfile = async (profileId: string) => {
        "use server";
        const profile = profiles.find((p) => p.id === profileId);
        if (!profile) {
            return { profile: {} as any, orders: [], favorites: [] };
```

```
        }

        const [orders, favorites] = await Promise.all([
            fetchProfileOrders(profileId),
            fetchProfileFavorites(profileId),
        ]);

        return { profile, orders, favorites };
    };

    return (
        <DashboardShell
            title="User profiles"
            description="Manage customer accounts, view order history, and track
loyalty status."
        >
            <ProfilesBoard profiles={profiles} onViewProfile={handleViewProfile}
/>
        </DashboardShell>
    );
}
</file>

<file path="admin/src/app/(dashboard)/settings/page.tsx">
import { DashboardShell } from "@/components/layout/DashboardShell";

export default function SettingsPage() {
  return (
    <DashboardShell
      title="Workspace settings"
      description="Configure notification channels and escalation defaults."
    >
      <div className="space-y-4 text-sm text-yellow-300">
        <div className="rounded-2xl border border-white/10 bg-yellow-900/40
p-4">
          <p className="text-xs uppercase tracking-[0.3em] text-yellow-500">
            Notifications
          </p>
          <p className="mt-2 text-white">
            Ops alerts sent to #war-room (Teams)
          </p>
          <p>
            Email digests go to ops@pacatering.co.za every morning at 06:00.
          </p>
        </div>
        <div className="rounded-2xl border border-white/10 bg-yellow-900/40
p-4">
          <p className="text-xs uppercase tracking-[0.3em] text-yellow-500">
            Escalations
          </p>
          <p className="mt-2 text-white">On-call: Nomsa Nkosi</p>
          <p>Backup: Lesedi Moagi</p>
        </div>
      </div>
    </DashboardShell>
  );
}
</file>

<file path="admin/src/app/(dashboard)/support/page.tsx">
import { DashboardShell } from "@/components/layout/DashboardShell";

export default function SupportPage() {
  return (
```

```
    <DashboardShell
      title="Ops support"
      description="Get help from leadership or open a ticket with the tech
team."
    >
      <div className="space-y-4 text-sm text-yellow-300">
        <p>Need assistance? Reach out via the following channels:</p>
        <ul className="list-disc space-y-2 pl-5">
          <li>Email: ops@pacatering.co.za</li>
          <li>WhatsApp: +27 60 555 0101</li>
          <li>Escalations: #ops-war-room (Teams)</li>
        </ul>
      </div>
    </DashboardShell>
  );
}
</file>

<file path="admin/src/app/layout.tsx">
import type { Metadata } from "next";
import { Geist, Geist_Mono } from "next/font/google";
import "./globals.css";

const geistSans = Geist({
  variable: "--font-geist-sans",
  subsets: ["latin"],
});

const geistMono = Geist_Mono({
  variable: "--font-geist-mono",
  subsets: ["latin"],
});

export const metadata: Metadata = {
  title: "PA Catering Admin",
  description: "Operations console for menu, orders, and logistics",
};

export default function RootLayout({
  children,
}: Readonly<{
  children: React.ReactNode;
}>) {
  return (
    <html lang="en">
      <body
        className={`${geistSans.variable} ${geistMono.variable} antialiased bg-
yellow-950 text-white`}
      >
        {children}
      </body>
    </html>
  );
}
</file>

<file path="admin/src/components/auth/LoginForm.tsx">
"use client";

import { useFormState } from "react-dom";
import { SubmitButton } from "@/components/forms/SubmitButton";
import type { AuthActionState } from "@/lib/auth";
import { loginAction } from "@/lib/auth";
```

```
type Props = {
  redirectTo?: string;
  initialMessage?: string;
};

const initialState: AuthActionState = {};

export const LoginForm = ({ redirectTo, initialMessage }: Props) => {
  const [state, formAction] = useFormState(loginAction, {
    ...initialState,
    message: initialMessage,
  });

  return (
    <form action={formAction} className="space-y-4">
      <input
        type="hidden"
        name="redirectTo"
        value={redirectTo ?? "/dashboard"}
      />
      <div className="space-y-2">
        <label className="text-sm text-yellow-300" htmlFor="email">
          Work email
        </label>
        <input
          id="email"
          name="email"
          type="email"
          required
          className="w-full rounded-lg border border-white/10 bg-yellow-900/60
px-4 py-2 text-white outline-none transition focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
          placeholder="you@pacatering.co.za"
        />
      </div>

      <div className="space-y-2">
        <label className="text-sm text-yellow-300" htmlFor="password">
          Password
        </label>
        <input
          id="password"
          name="password"
          type="password"
          required
          className="w-full rounded-lg border border-white/10 bg-yellow-900/60
px-4 py-2 text-white outline-none transition focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
          placeholder="â ¢â ¢â ¢â ¢â ¢â ¢â ¢â ¢"
        />
      </div>

      {(state.error || state.message) && (
        <p
          className={`text-sm ${state.error ? "text-rose-400" : "text-
emerald-400"}`}
        >
          {state.error ?? state.message}
        </p>
      )}

      <SubmitButton label="Sign in" loadingLabel="Signing in..." />
    </form>
  );
```

```
};
</file>

<file path="admin/src/components/auth/RegisterForm.tsx">
"use client";

import { useFormState } from "react-dom";
import { SubmitButton } from "@/components/forms/SubmitButton";
import type { AuthActionState } from "@/lib/auth";
import { registerAction } from "@/lib/auth";

type Props = {
  initialError?: string;
};

export const RegisterForm = ({ initialError }: Props) => {
  const [state, formAction] = useFormState<AuthActionState, FormData>(
    registerAction,
    {
      error: initialError,
    },
  );

  return (
    <form action={formAction} className="space-y-4">
      <div className="space-y-2">
        <label className="text-sm text-yellow-300" htmlFor="fullName">
          Full name
        </label>
        <input
          id="fullName"
          name="fullName"
          type="text"
          placeholder="Nomsa Nkosi"
          className="w-full rounded-lg border border-white/10 bg-yellow-900/60
px-4 py-2 text-white outline-none transition focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
        />
      </div>

      <div className="space-y-2">
        <label className="text-sm text-yellow-300" htmlFor="email">
          Work email
        </label>
        <input
          id="email"
          name="email"
          type="email"
          required
          placeholder="you@pacatering.co.za"
          className="w-full rounded-lg border border-white/10 bg-yellow-900/60
px-4 py-2 text-white outline-none transition focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
        />
      </div>

      <div className="space-y-2">
        <label className="text-sm text-yellow-300" htmlFor="password">
          Password
        </label>
        <input
          id="password"
          name="password"
          type="password"
```

```
          required
          minLength={6}
          placeholder="Create a strong password"
          className="w-full rounded-lg border border-white/10 bg-yellow-900/60
px-4 py-2 text-white outline-none transition focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
        />
      </div>

      <div className="space-y-2">
        <label className="text-sm text-yellow-300" htmlFor="inviteCode">
          Invite code
        </label>
        <input
          id="inviteCode"
          name="inviteCode"
          type="password"
          placeholder="Required if set by Ops"
          className="w-full rounded-lg border border-white/10 bg-yellow-900/60
px-4 py-2 text-white outline-none transition focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
        />
      </div>

      {state.error ? (
        <p className="text-sm text-rose-400">{state.error}</p>
      ) : null}

      <SubmitButton
        label="Create admin account"
        loadingLabel="Creating account..."
      />
    </form>
  );
};
</file>

<file path="admin/src/components/contact/ContactBoard.tsx">
"use client";

import { useState } from "react";
import type { ContactRecord } from "@/lib/types";
import { formatDistanceToNow } from "date-fns";

type Props = {
    contacts: ContactRecord[];
};

export const ContactBoard = ({ contacts }: Props) => {
    const [search, setSearch] = useState("");
    const [selectedContact, setSelectedContact] = useState<ContactRecord |
null>(
        null,
    );

    const filtered = contacts.filter(
        (contact) =>
            !search ||
            contact.name.toLowerCase().includes(search.toLowerCase()) ||
            contact.email.toLowerCase().includes(search.toLowerCase()) ||
            contact.message.toLowerCase().includes(search.toLowerCase()),
    );

    return (
```

```
        <div className="space-y-4">
            <div className="flex items-center justify-between gap-3">
                <input
                    type="search"
                    placeholder="Search contact submissions..."
                    value={search}
                    onChange={(e) => setSearch(e.target.value)}
                    className="w-full max-w-md rounded-full border border-
white/10 bg-yellow-900/60 px-4 py-3 text-white outline-none focus:border-
indigo-400 focus:ring-2 focus:ring-indigo-400/40"
                />
                <p className="text-xs uppercase tracking-[0.3em] text-
yellow-500">
                    {filtered.length} submissions
                </p>
            </div>

            <div className="grid gap-4 md:grid-cols-2 lg:grid-cols-3">
                {filtered.map((contact) => (
                    <div
                        key={contact.id}
                        onClick={() => setSelectedContact(contact)}
                        className="cursor-pointer rounded-2xl border border-
white/10 bg-yellow-900/40 p-4 transition-colors hover:bg-yellow-900/60"
                    >
                        <div className="flex items-start justify-between">
                            <div>
                                <h3 className="font-semibold text-
white">{contact.name}</h3>
                                <p className="text-sm text-
yellow-400">{contact.email}</p>
                                {contact.phone && (
                                    <p className="text-sm text-
yellow-500">{contact.phone}</p>
                                )}
                            </div>
                            <p className="text-xs text-yellow-500">
                                {formatDistanceToNow(new
Date(contact.created_at), {
                                    addSuffix: true,
                                })}
                            </p>
                        </div>
                        <p className="mt-3 line-clamp-2 text-sm text-
yellow-300">
                            {contact.message}
                        </p>
                    </div>
                ))}
            </div>

            {selectedContact && (
                <ContactDetailModal
                    contact={selectedContact}
                    onClose={() => setSelectedContact(null)}
                />
            )}
        </div>
    );
};

type ContactDetailModalProps = {
    contact: ContactRecord;
    onClose: () => void;
```

```
};

const ContactDetailModal = ({ contact, onClose }: ContactDetailModalProps) => {
    return (
        <div className="fixed inset-0 z-50 flex items-center justify-center bg-
black/70">
            <div className="w-full max-w-2xl rounded-2xl border border-white/10
bg-yellow-900 p-6 shadow-2xl">
                <div className="flex items-start justify-between">
                    <div>
                        <h3 className="text-xl font-semibold text-
white">{contact.name}</h3>
                        <p className="text-sm text-yellow-400">{contact.email}
</p>
                        {contact.phone && (
                            <p className="text-sm text-
yellow-400">{contact.phone}</p>
                        )}
                        <p className="mt-1 text-xs text-yellow-500">
                            Submitted{" "}
                            {formatDistanceToNow(new Date(contact.created_at), {
                                addSuffix: true,
                            })}
                        </p>
                    </div>
                    <button
                        type="button"
                        onClick={onClose}
                        className="rounded-lg bg-yellow-800 px-4 py-2 text-sm
text-white hover:bg-yellow-700"
                    >
                        Close
                    </button>
                </div>

                <div className="mt-6">
                    <p className="text-xs uppercase tracking-[0.3em] text-
yellow-500">
                        Message
                    </p>
                    <div className="mt-2 rounded-xl border border-white/10 bg-
yellow-950/40 p-4">
                        <p className="whitespace-pre-wrap text-sm text-
yellow-200">
                            {contact.message}
                        </p>
                    </div>
                </div>
            </div>
        </div>
    );
};
</file>

<file path="admin/src/components/dashboard/RecentOrdersCard.tsx">
import { formatDistanceToNow } from "date-fns";
import Link from "next/link";
import {
  HiOutlineArrowRight,
  HiOutlineClipboardDocumentCheck,
} from "react-icons/hi2";
import { DashboardShell } from "@/components/layout/DashboardShell";
import type { SupabaseOrderRecord } from "@/lib/types";
```

```
type Props = {
  orders: SupabaseOrderRecord[];
};

const statusColor: Record<string, string> = {
  pending: "text-amber-300",
  processing: "text-sky-300",
  completed: "text-emerald-300",
  cancelled: "text-rose-300",
};

export const RecentOrdersCard = ({ orders }: Props) => (
  <DashboardShell
    title="Live orders"
    description="Monitor field teams and respond to escalations in real time."
    actions={
      <Link
        href="/orders"
        className="inline-flex items-center gap-1 rounded-full border border-
white/10 px-3 py-2 text-xs uppercase tracking-[0.25em] text-yellow-200
transition hover:border-indigo-400 hover:text-white"
      >
        View queue
        <HiOutlineArrowRight />
      </Link>
    }
  >
    <div className="overflow-x-auto">
      <table className="min-w-full text-left text-sm text-yellow-300">
        <thead className="text-xs uppercase tracking-[0.2em] text-yellow-500">
          <tr>
            <th className="py-3">Order</th>
            <th className="py-3">Status</th>
            <th className="py-3">Total</th>
            <th className="py-3">Updated</th>
          </tr>
        </thead>
        <tbody>
          {orders.length === 0 ? (
            <tr>
              <td colSpan={4} className="py-10 text-center text-yellow-500">
                <div className="flex flex-col items-center gap-2">
                  <HiOutlineClipboardDocumentCheck className="text-3xl text-
yellow-600" />
                  <p>No orders in the last hour.</p>
                </div>
              </td>
            </tr>
          ) : (
            orders.map((order) => (
              <tr key={order.id} className="border-t border-white/5">
                <td className="py-3 font-semibold text-white">
                  #{order.id.slice(0, 8)}
                </td>
                <td
                  className={`py-3 text-xs uppercase tracking-[0.25em] $
{statusColor[order.status] ?? "text-yellow-400"}`}
                >
                  {order.status}
                </td>
                <td className="py-3 font-medium text-white">
                  R{order.total_price?.toFixed(2) ?? "0.00"}
                </td>
                <td className="py-3 text-yellow-400">
```

```
                  {formatDistanceToNow(new Date(order.created_at), {
                    addSuffix: true,
                  })}
                </td>
              </tr>
            ))
          )}
        </tbody>
      </table>
    </div>
  </DashboardShell>
);
</file>

<file path="admin/src/components/dashboard/RevenueTrendCard.tsx">
import { HiOutlineArrowTrendingUp } from "react-icons/hi2";
import { DashboardShell } from "@/components/layout/DashboardShell";
import type { RevenuePoint } from "@/lib/data/dashboard";

type Props = {
  series: RevenuePoint[];
};

export const RevenueTrendCard = ({ series }: Props) => (
  <DashboardShell
    title="7-day revenue trend"
    description="Daily gross sales captured via Supabase orders API."
  >
    {series.length === 0 ? (
      <div className="flex h-32 items-center justify-center text-sm text-
yellow-500">
        No orders to chart yet.
      </div>
    ) : (
      <ul className="space-y-3">
        {series.map((point) => (
          <li
            key={point.date}
            className="flex items-center justify-between rounded-xl border
border-white/5 bg-yellow-900/40 px-3 py-2 text-sm"
          >
            <div>
              <p className="text-xs uppercase tracking-[0.3em] text-yellow-500">
                {point.date}
              </p>
              <p className="text-lg font-semibold text-white">
                R{point.total.toFixed(2)}
              </p>
            </div>
            <HiOutlineArrowTrendingUp className="text-xl text-emerald-300" />
          </li>
        ))}
      </ul>
    )}
  </DashboardShell>
);
</file>

<file path="admin/src/components/dashboard/StatsGrid.tsx">
import {
  HiOutlineArrowTrendingUp,
  HiOutlineBanknotes,
  HiOutlineClipboardDocumentList,
  HiOutlineCube,
```

```
  HiOutlineSparkles,
} from "react-icons/hi2";
import type { DashboardStats } from "@/lib/data/dashboard";

type Props = {
  stats: DashboardStats;
};

const formatNumber = (value: number) => value.toLocaleString("en-ZA");

export const StatsGrid = ({ stats }: Props) => {
  const cards = [
    {
      label: "Orders (24h)",
      value: formatNumber(stats.totalOrders),
      change: "+12% vs yesterday",
      icon: HiOutlineClipboardDocumentList,
    },
    {
      label: "Revenue (ZAR)",
      value: `R${formatNumber(stats.revenue)}`,
      change: "Gross sales",
      icon: HiOutlineBanknotes,
    },
    {
      label: "Pending orders",
      value: formatNumber(stats.pendingOrders),
      change: "Awaiting assignment",
      icon: HiOutlineArrowTrendingUp,
    },
    {
      label: "Menu items live",
      value: formatNumber(stats.productCount),
      change: `${stats.lowStockCount} flagged low stock`,
      icon: HiOutlineCube,
    },
    {
      label: "Fulfilment rate",
      value: `${stats.fulfillmentRate}%`,
      change: "Completed vs total",
      icon: HiOutlineSparkles,
    },
  ];

  return (
    <div className="grid gap-4 md:grid-cols-2 xl:grid-cols-3 2xl:grid-cols-5">
      {cards.map(({ label, value, change, icon: Icon }) => (
        <article
          key={label}
          className="rounded-2xl border border-white/5 bg-gradient-to-br from-
yellow-900/70 to-yellow-900/30 p-4 shadow-lg shadow-black/10"
        >
          <div className="flex items-center justify-between">
            <div>
              <p className="text-sm uppercase tracking-[0.2em] text-yellow-400">
                {label}
              </p>
              <p className="mt-2 text-2xl font-semibold text-white">{value}</p>
            </div>
            <div className="rounded-xl bg-white/10 p-3 text-indigo-300">
              <Icon className="text-xl" />
            </div>
          </div>
          <p className="mt-4 text-xs uppercase tracking-[0.25em] text-
```

```
yellow-500">
            {change}
          </p>
        </article>
      ))}
    </div>
  );
};
</file>

<file path="admin/src/components/layout/DashboardShell.tsx">
import type { ReactNode } from "react";

type Props = {
  title: string;
  description?: string;
  actions?: ReactNode;
  children: ReactNode;
};

export const DashboardShell = ({
  title,
  description,
  actions,
  children,
}: Props) => (
  <section className="space-y-6 rounded-2xl border border-white/5 bg-
yellow-900/40 p-6 shadow-inner shadow-black/20">
    <header className="flex flex-col gap-2 border-b border-white/5 pb-4 md:flex-
row md:items-center md:justify-between">
      <div>
        <h2 className="text-xl font-semibold text-white">{title}</h2>
        {description ? (
          <p className="text-sm text-yellow-400">{description}</p>
        ) : null}
      </div>
      {actions ? (
        <div className="flex items-center gap-3">{actions}</div>
      ) : null}
    </header>

    <div>{children}</div>
  </section>
);
</file>

<file path="admin/src/components/menu/comments/CommentsBoard.tsx">
"use client";

import { useState } from "react";
import { useFormState } from "react-dom";
import { SubmitButton } from "@/components/forms/SubmitButton";
import { DeleteConfirmDialog } from "@/components/shared/DeleteConfirmDialog";
import {
    type CommentActionState,
    updateCommentAction,
    deleteCommentAction,
} from "@/lib/data/comments-actions";
import type { CommentRecord } from "@/lib/types";
import { formatDistanceToNow } from "date-fns";

type Props = {
    comments: CommentRecord[];
};
```

```
const initialState: CommentActionState = {};

export const CommentsBoard = ({ comments }: Props) => {
    const [search, setSearch] = useState("");
    const [editingId, setEditingId] = useState<string | null>(null);
    const [deleteId, setDeleteId] = useState<string | null>(null);

    const filtered = comments.filter(
        (comment) =>
            !search ||
            comment.body.toLowerCase().includes(search.toLowerCase()) ||
            comment.user_name?.toLowerCase().includes(search.toLowerCase()),
    );

    const handleDelete = async () => {
        if (deleteId) {
            await deleteCommentAction(deleteId);
            setDeleteId(null);
        }
    };

    return (
        <div className="space-y-4">
            <div className="flex items-center justify-between gap-3">
                <input
                    type="search"
                    placeholder="Search comments..."
                    value={search}
                    onChange={(e) => setSearch(e.target.value)}
                    className="w-full max-w-md rounded-full border border-
white/10 bg-yellow-900/60 px-4 py-3 text-white outline-none focus:border-
indigo-400 focus:ring-2 focus:ring-indigo-400/40"
                />
                <p className="text-xs uppercase tracking-[0.3em] text-
yellow-500">
                    {filtered.length} comments
                </p>
            </div>

            <div className="space-y-3">
                {filtered.map((comment) => (
                    <CommentCard
                        key={comment.id}
                        comment={comment}
                        isEditing={editingId === comment.id}
                        onEdit={() => setEditingId(comment.id)}
                        onCancelEdit={() => setEditingId(null)}
                        onDelete={() => setDeleteId(comment.id)}
                    />
                ))}
            </div>

            <DeleteConfirmDialog
                isOpen={deleteId !== null}
                onClose={() => setDeleteId(null)}
                onConfirm={handleDelete}
                title="Delete Comment"
                message="Are you sure you want to delete this comment? This
action cannot be undone."
            />
        </div>
    );
};
```

```
type CommentCardProps = {
    comment: CommentRecord;
    isEditing: boolean;
    onEdit: () => void;
    onCancelEdit: () => void;
    onDelete: () => void;
};

const CommentCard = ({
    comment,
    isEditing,
    onEdit,
    onCancelEdit,
    onDelete,
}: CommentCardProps) => {
    const [state, formAction] = useFormState(updateCommentAction, initialState);

    return (
        <div className="rounded-2xl border border-white/10 bg-yellow-900/40
p-4">
            <div className="flex items-start justify-between gap-4">
                <div className="flex-1">
                    <div className="flex items-center gap-2">
                        <p className="text-sm font-semibold text-white">
                            {comment.user_name || "Anonymous"}
                        </p>
                        <p className="text-xs text-yellow-500">
                            {formatDistanceToNow(new Date(comment.created_at), {
                                addSuffix: true,
                            })}
                        </p>
                    </div>

                    {isEditing ? (
                        <form action={formAction} className="mt-3 space-y-3">
                            <input type="hidden" name="id" value={comment.id} />
                            <textarea
                                name="body"
                                defaultValue={comment.body}
                                rows={3}
                                className="w-full rounded-lg border border-
white/10 bg-yellow-900/60 px-3 py-2 text-white outline-none focus:border-
indigo-400 focus:ring-2 focus:ring-indigo-400/40"
                            />
                            <input
                                name="user_name"
                                defaultValue={comment.user_name || ""}
                                placeholder="User name"
                                className="w-full rounded-lg border border-
white/10 bg-yellow-900/60 px-3 py-2 text-sm text-white outline-none
focus:border-indigo-400 focus:ring-2 focus:ring-indigo-400/40"
                            />
                            {state.error && (
                                <p className="text-sm text-
rose-400">{state.error}</p>
                            )}
                            {state.success && (
                                <p className="text-sm text-
emerald-400">{state.success}</p>
                            )}
                            <div className="flex gap-2">
                                <SubmitButton
                                    label="Save"
```

```
                                    loadingLabel="Saving..."
                                    className="bg-indigo-600 px-4 py-2 text-sm"
                                />
                                <button
                                    type="button"
                                    onClick={onCancelEdit}
                                    className="rounded-lg bg-yellow-800 px-4
py-2 text-sm text-white hover:bg-yellow-700"
                                >
                                    Cancel
                                </button>
                            </div>
                        </form>
                    ) : (
                        <p className="mt-2 text-sm text-
yellow-300">{comment.body}</p>
                    )}
                </div>

                {!isEditing && (
                    <div className="flex gap-2">
                        <button
                            type="button"
                            onClick={onEdit}
                            className="rounded-lg bg-yellow-800 px-3 py-1 text-
xs text-white hover:bg-yellow-700"
                        >
                            Edit
                        </button>
                        <button
                            type="button"
                            onClick={onDelete}
                            className="rounded-lg bg-rose-600 px-3 py-1 text-xs
text-white hover:bg-rose-700"
                        >
                            Delete
                        </button>
                    </div>
                )}
            </div>
        </div>
    );
};
</file>

<file path="admin/src/components/menu/favorites/FavoritesBoard.tsx">
"use client";

import { useState } from "react";
import type { UserFavoriteRecord } from "@/lib/types";
import { formatDistanceToNow } from "date-fns";

type Props = {
    favorites: UserFavoriteRecord[];
};

export const FavoritesBoard = ({ favorites }: Props) => {
    const [search, setSearch] = useState("");

    const filtered = favorites.filter(
        (fav) =>
            !search ||
            fav.user_id.toLowerCase().includes(search.toLowerCase()) ||
            fav.product_id.toLowerCase().includes(search.toLowerCase()),
```

```
    );

    return (
        <div className="space-y-4">
            <div className="flex items-center justify-between gap-3">
                <input
                    type="search"
                    placeholder="Search favorites..."
                    value={search}
                    onChange={(e) => setSearch(e.target.value)}
                    className="w-full max-w-md rounded-full border border-
white/10 bg-yellow-900/60 px-4 py-3 text-white outline-none focus:border-
indigo-400 focus:ring-2 focus:ring-indigo-400/40"
                />
                <p className="text-xs uppercase tracking-[0.3em] text-
yellow-500">
                    {filtered.length} favorites
                </p>
            </div>

            <div className="rounded-2xl border border-white/10 bg-
yellow-900/40">
                <table className="w-full text-sm">
                    <thead>
                        <tr className="border-b border-white/10 text-left">
                            <th className="p-4 font-medium text-yellow-400">User
ID</th>
                            <th className="p-4 font-medium text-
yellow-400">Product ID</th>
                            <th className="p-4 font-medium text-
yellow-400">Created</th>
                        </tr>
                    </thead>
                    <tbody>
                        {filtered.map((favorite, index) => (
                            <tr
                                key={`${favorite.user_id}-${favorite.product_id}
`}
                                className={`${index !== filtered.length - 1
                                    ? "border-b border-white/10"
                                    : ""
                                    }`}
                            >
                                <td className="p-4 font-mono text-xs text-
yellow-300">
                                    {favorite.user_id.slice(0, 8)}...
                                </td>
                                <td className="p-4 font-mono text-xs text-
yellow-300">
                                    {favorite.product_id.slice(0, 8)}...
                                </td>
                                <td className="p-4 text-yellow-400">
                                    {formatDistanceToNow(new
Date(favorite.created_at), {
                                        addSuffix: true,
                                    })}
                                </td>
                            </tr>
                        ))}
                    </tbody>
                </table>
            </div>
        </div>
    );
```

```
};
</file>

<file path="admin/src/components/orders/OrdersBoard.tsx">
"use client";

import { formatDistanceToNow } from "date-fns";
import { useMemo, useState } from "react";
import type { CartItem, SupabaseOrderRecord } from "@/lib/types";
import { UpdateOrderStatusForm } from "./UpdateOrderStatusForm";

type Props = {
  orders: SupabaseOrderRecord[];
};

const STATUS_COLUMNS: SupabaseOrderRecord["status"][] = [
  "pending",
  "processing",
  "completed",
  "cancelled",
];

type DisplayOrderItem = CartItem & {
  name?: string;
  Quantity?: number;
  ProductID?: string;
};

const resolveItemName = (item: DisplayOrderItem) =>
  item.Name ?? item.name ?? "Menu item";

const resolveItemQuantity = (item: DisplayOrderItem) =>
  item.quantity ?? item.Quantity ?? 1;

export const OrdersBoard = ({ orders }: Props) => {
  const [search, setSearch] = useState("");

  const filtered = useMemo(() => {
    return orders.filter((order) => {
      if (!search) return true;
      const term = search.toLowerCase();
      const idMatch = order.id.toLowerCase().includes(term);
      const userMatch = (order.user_id ?? "").toLowerCase().includes(term);
      const itemMatch = (order.items ?? []).some((item) =>
        resolveItemName(item as DisplayOrderItem)
          .toLowerCase()
          .includes(term),
      );
      return idMatch || userMatch || itemMatch;
    });
  }, [orders, search]);

  const grouped = STATUS_COLUMNS.map((status) => ({
    status,
    items: filtered.filter((order) => order.status === status),
  }));

  return (
    <div className="space-y-4">
      <div className="flex flex-col gap-3 md:flex-row md:items-center
md:justify-between">
        <label className="w-full max-w-md text-sm text-yellow-300">
          <span className="sr-only">Search orders</span>
          <input
```

```
            type="search"
            placeholder="Search by order ID, customer, or item"
            value={search}
            onChange={(event) => setSearch(event.target.value)}
            className="w-full rounded-full border border-white/10 bg-
yellow-900/60 px-4 py-3 text-white outline-none focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
          />
        </label>
        <p className="text-xs uppercase tracking-[0.3em] text-yellow-500">
          Showing {filtered.length} of {orders.length} orders
        </p>
      </div>

      <div className="grid gap-4 md:grid-cols-2 xl:grid-cols-4">
        {grouped.map(({ status, items }) => (
          <section
            key={status}
            className="rounded-2xl border border-white/10 bg-yellow-900/40 p-4"
          >
            <header className="flex items-center justify-between">
              <p className="text-xs uppercase tracking-[0.3em] text-yellow-500">
                {status}
              </p>
              <span className="rounded-full bg-white/10 px-3 py-1 text-xs text-
white">
                {items.length}
              </span>
            </header>

            <div className="mt-4 space-y-3">
              {items.length === 0 ? (
                <p className="text-xs text-yellow-500">
                  No orders in this lane.
                </p>
              ) : (
                items.map((order) => (
                  <article
                    key={order.id}
                    className="rounded-xl border border-white/10 bg-
yellow-950/40 p-3 text-sm text-yellow-200"
                  >
                    <div className="flex items-start justify-between gap-2">
                      <div>
                        <p className="text-xs uppercase tracking-[0.3em] text-
yellow-500">
                          #{order.id.slice(0, 8)}
                        </p>
                        <p className="font-semibold text-white">
                          R{order.total_price?.toFixed(2) ?? "0.00"}
                        </p>
                        <p className="text-xs text-yellow-500">
                          {formatDistanceToNow(new Date(order.created_at), {
                            addSuffix: true,
                          })}
                        </p>
                      </div>
                      <div className="text-right text-xs text-yellow-500">
                        <p>{order.total_quantity} items</p>
                        <p>{order.user_id ?? "Guest"}</p>
                      </div>
                    </div>

                    <div className="mt-3 rounded-lg bg-yellow-900/60 p-2 text-xs
```

```
text-yellow-400">
                      <p className="text-[10px] uppercase tracking-[0.4em] text-
yellow-500">
                        Items
                      </p>
                      <ul className="mt-1 space-y-1">
                        {(order.items ?? []).slice(0, 4).map((item, index) => {
                          const displayItem = item as DisplayOrderItem;
                          const name = resolveItemName(displayItem);
                          const quantity = resolveItemQuantity(displayItem);
                          return (
                            <li
                              key={`${order.id}-${displayItem.ProductID ??
name}-${index}`}
                              className="flex justify-between text-white"
                            >
                              <span>{name}</span>
                              <span className="text-yellow-400">
                                x{quantity}
                              </span>
                            </li>
                          );
                        })}
                        {(order.items ?? []).length > 4 ? (
                          <li className="text-[10px] uppercase tracking-[0.3em]
text-yellow-500">
                            + more
                          </li>
                        ) : null}
                      </ul>
                    </div>

                    <div className="mt-3">
                      <UpdateOrderStatusForm
                        orderId={order.id}
                        currentStatus={order.status}
                      />
                    </div>
                  </article>
                ))
              )}
            </div>
          </section>
        ))}
      </div>
    </div>
  );
};
</file>

<file path="admin/src/components/photo-bookings/PhotoBookingsBoard.tsx">
"use client";

import { useState } from "react";
import { useFormState } from "react-dom";
import { SubmitButton } from "@/components/forms/SubmitButton";
import {
    type PhotoBookingActionState,
    updatePhotoBookingAction,
} from "@/lib/data/photo-bookings-actions";
import type { PhotoBookingRecord } from "@/lib/types";
import { format } from "date-fns";

type Props = {
```

```
    bookings: PhotoBookingRecord[];
};

const initialState: PhotoBookingActionState = {};

export const PhotoBookingsBoard = ({ bookings }: Props) => {
    const [search, setSearch] = useState("");
    const [editingId, setEditingId] = useState<string | null>(null);

    const filtered = bookings.filter(
        (booking) =>
            !search ||
            booking.name.toLowerCase().includes(search.toLowerCase()) ||
            booking.email.toLowerCase().includes(search.toLowerCase()) ||
            booking.package.toLowerCase().includes(search.toLowerCase()),
    );

    return (
        <div className="space-y-4">
            <div className="flex items-center justify-between gap-3">
                <input
                    type="search"
                    placeholder="Search bookings..."
                    value={search}
                    onChange={(e) => setSearch(e.target.value)}
                    className="w-full max-w-md rounded-full border border-
white/10 bg-yellow-900/60 px-4 py-3 text-white outline-none focus:border-
indigo-400 focus:ring-2 focus:ring-indigo-400/40"
                />
                <p className="text-xs uppercase tracking-[0.3em] text-
yellow-500">
                    {filtered.length} bookings
                </p>
            </div>

            <div className="grid gap-4 md:grid-cols-2">
                {filtered.map((booking) => (
                    <BookingCard
                        key={booking.id}
                        booking={booking}
                        isEditing={editingId === booking.id}
                        onEdit={() => setEditingId(booking.id)}
                        onCancelEdit={() => setEditingId(null)}
                    />
                ))}
            </div>
        </div>
    );
};

type BookingCardProps = {
    booking: PhotoBookingRecord;
    isEditing: boolean;
    onEdit: () => void;
    onCancelEdit: () => void;
};

const BookingCard = ({
    booking,
    isEditing,
    onEdit,
    onCancelEdit,
}: BookingCardProps) => {
    const [state, formAction] = useFormState(
```

```
        updatePhotoBookingAction,
        initialState,
    );

    return (
        <div className="rounded-2xl border border-white/10 bg-yellow-900/40
p-4">
            {isEditing ? (
                <form action={formAction} className="space-y-3">
                    <input type="hidden" name="id" value={booking.id} />

                    <div className="grid gap-3 md:grid-cols-2">
                        <label className="space-y-2 text-sm">
                            <span className="text-yellow-300">Name</span>
                            <input
                                name="name"
                                defaultValue={booking.name}
                                required
                                className="w-full rounded-lg border border-
white/10 bg-yellow-900/60 px-3 py-2 text-white outline-none focus:border-
indigo-400 focus:ring-2 focus:ring-indigo-400/40"
                            />
                        </label>
                        <label className="space-y-2 text-sm">
                            <span className="text-yellow-300">Email</span>
                            <input
                                name="email"
                                type="email"
                                defaultValue={booking.email}
                                required
                                className="w-full rounded-lg border border-
white/10 bg-yellow-900/60 px-3 py-2 text-white outline-none focus:border-
indigo-400 focus:ring-2 focus:ring-indigo-400/40"
                            />
                        </label>
                    </div>

                    <div className="grid gap-3 md:grid-cols-2">
                        <label className="space-y-2 text-sm">
                            <span className="text-yellow-300">Phone</span>
                            <input
                                name="phone"
                                defaultValue={booking.phone || ""}
                                className="w-full rounded-lg border border-
white/10 bg-yellow-900/60 px-3 py-2 text-white outline-none focus:border-
indigo-400 focus:ring-2 focus:ring-indigo-400/40"
                            />
                        </label>
                        <label className="space-y-2 text-sm">
                            <span className="text-yellow-300">Package</span>
                            <input
                                name="package"
                                defaultValue={booking.package}
                                required
                                className="w-full rounded-lg border border-
white/10 bg-yellow-900/60 px-3 py-2 text-white outline-none focus:border-
indigo-400 focus:ring-2 focus:ring-indigo-400/40"
                            />
                        </label>
                    </div>

                    <div className="grid gap-3 md:grid-cols-3">
                        <label className="space-y-2 text-sm">
                            <span className="text-yellow-300">Date</span>
```

```
                            <input
                                name="date"
                                type="date"
                                defaultValue={booking.date}
                                required
                                className="w-full rounded-lg border border-
white/10 bg-yellow-900/60 px-3 py-2 text-white outline-none focus:border-
indigo-400 focus:ring-2 focus:ring-indigo-400/40"
                            />
                        </label>
                        <label className="space-y-2 text-sm">
                            <span className="text-yellow-300">Time</span>
                            <input
                                name="time"
                                type="time"
                                defaultValue={booking.time}
                                required
                                className="w-full rounded-lg border border-
white/10 bg-yellow-900/60 px-3 py-2 text-white outline-none focus:border-
indigo-400 focus:ring-2 focus:ring-indigo-400/40"
                            />
                        </label>
                        <label className="space-y-2 text-sm">
                            <span className="text-yellow-300">People</span>
                            <input
                                name="people"
                                type="number"
                                min="1"
                                defaultValue={booking.people}
                                required
                                className="w-full rounded-lg border border-
white/10 bg-yellow-900/60 px-3 py-2 text-white outline-none focus:border-
indigo-400 focus:ring-2 focus:ring-indigo-400/40"
                            />
                        </label>
                    </div>

                    <label className="block space-y-2 text-sm">
                        <span className="text-yellow-300">Message</span>
                        <textarea
                            name="message"
                            rows={2}
                            defaultValue={booking.message || ""}
                            className="w-full rounded-lg border border-white/10
bg-yellow-900/60 px-3 py-2 text-white outline-none focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
                        />
                    </label>

                    {state.error && <p className="text-sm text-
rose-400">{state.error}</p>}
                    {state.success && (
                        <p className="text-sm text-emerald-400">{state.success}
</p>
                    )}

                    <div className="flex gap-2">
                        <SubmitButton
                            label="Save"
                            loadingLabel="Saving..."
                            className="bg-indigo-600 px-4 py-2 text-sm"
                        />
                        <button
                            type="button"
```

```
                            onClick={onCancelEdit}
                            className="rounded-lg bg-yellow-800 px-4 py-2 text-
sm text-white hover:bg-yellow-700"
                        >
                            Cancel
                        </button>
                    </div>
                </form>
            ) : (
                <>
                    <div className="flex items-start justify-between">
                        <div>
                            <h3 className="font-semibold text-
white">{booking.name}</h3>
                            <p className="text-sm text-
yellow-400">{booking.email}</p>
                            {booking.phone && (
                                <p className="text-sm text-
yellow-500">{booking.phone}</p>
                            )}
                        </div>
                        <button
                            type="button"
                            onClick={onEdit}
                            className="rounded-lg bg-yellow-800 px-3 py-1 text-
xs text-white hover:bg-yellow-700"
                        >
                            Edit
                        </button>
                    </div>

                    <div className="mt-4 grid gap-2 text-sm">
                        <div className="flex justify-between">
                            <span className="text-yellow-500">Package:</span>
                            <span className="font-medium text-
white">{booking.package}</span>
                        </div>
                        <div className="flex justify-between">
                            <span className="text-yellow-500">Date:</span>
                            <span className="text-white">
                                {format(new Date(booking.date), "PPP")}
                            </span>
                        </div>
                        <div className="flex justify-between">
                            <span className="text-yellow-500">Time:</span>
                            <span className="text-white">{booking.time}</span>
                        </div>
                        <div className="flex justify-between">
                            <span className="text-yellow-500">People:</span>
                            <span className="text-white">{booking.people}</span>
                        </div>
                    </div>

                    {booking.message && (
                        <div className="mt-3 rounded-lg bg-yellow-950/40 p-3">
                            <p className="text-xs uppercase tracking-[0.3em]
text-yellow-500">
                                Message
                            </p>
                            <p className="mt-1 text-sm text-
yellow-300">{booking.message}</p>
                        </div>
                    )}
                </>
```

```
            )}
        </div>
    );
};
</file>

<file path="admin/src/components/shared/DeleteConfirmDialog.tsx">
"use client";

import { useState } from "react";

type Props = {
    isOpen: boolean;
    onClose: () => void;
    onConfirm: () => void;
    title: string;
    message: string;
    confirmLabel?: string;
};

export const DeleteConfirmDialog = ({
    isOpen,
    onClose,
    onConfirm,
    title,
    message,
    confirmLabel = "Delete",
}: Props) => {
    const [isDeleting, setIsDeleting] = useState(false);

    if (!isOpen) return null;

    const handleConfirm = async () => {
        setIsDeleting(true);
        await onConfirm();
        setIsDeleting(false);
    };

    return (
        <div className="fixed inset-0 z-50 flex items-center justify-center bg-
black/70">
            <div className="w-full max-w-md rounded-2xl border border-white/10
bg-yellow-900 p-6 shadow-2xl">
                <h3 className="text-xl font-semibold text-white">{title}</h3>
                <p className="mt-3 text-sm text-yellow-300">{message}</p>

                <div className="mt-6 flex gap-3">
                    <button
                        type="button"
                        onClick={onClose}
                        disabled={isDeleting}
                        className="flex-1 rounded-lg border border-white/10 bg-
yellow-800 px-4 py-2 text-sm font-medium text-white hover:bg-yellow-700
disabled:opacity-50"
                    >
                        Cancel
                    </button>
                    <button
                        type="button"
                        onClick={handleConfirm}
                        disabled={isDeleting}
                        className="flex-1 rounded-lg bg-rose-600 px-4 py-2 text-
sm font-medium text-white hover:bg-rose-700 disabled:opacity-50"
                    >
```

```
                        {isDeleting ? "Deleting..." : confirmLabel}
                    </button>
                </div>
            </div>
        </div>
    );
};
</file>

<file path="admin/src/lib/data/comments-actions.ts">
"use server";

import { revalidatePath } from "next/cache";
import { createSupabaseServerClient } from "@/lib/supabase/server";

export type CommentActionState = {
    error?: string;
    success?: string;
};

const sanitize = (value: FormDataEntryValue | null) =>
    String(value ?? "").trim();

export const createCommentAction = async (
    _prev: CommentActionState,
    formData: FormData,
): Promise<CommentActionState> => {
    const productId = sanitize(formData.get("product_id"));
    const userId = sanitize(formData.get("user_id")) || null;
    const userName = sanitize(formData.get("user_name")) || null;
    const body = sanitize(formData.get("body"));

    if (!body) {
        return { error: "Comment body is required." };
    }

    const supabase = await createSupabaseServerClient();
    const commentData = {
        product_id: productId || null,
        user_id: userId,
        user_name: userName,
        body,
    };

    const { error } = await supabase.from("comments").insert(commentData as
never);

    if (error) {
        console.error("Failed to create comment", error);
        return { error: error.message };
    }

    revalidatePath("/comments");
    return { success: "Comment created successfully." };
};

export const updateCommentAction = async (
    _prev: CommentActionState,
    formData: FormData,
): Promise<CommentActionState> => {
    const id = sanitize(formData.get("id"));
    const body = sanitize(formData.get("body"));
    const userName = sanitize(formData.get("user_name")) || null;
```

```
    if (!id || !body) {
        return { error: "Comment ID and body are required." };
    }

    const supabase = await createSupabaseServerClient();
    const updateData = {
        body,
        user_name: userName,
    };

    const { error } = await supabase
        .from("comments")
        .update(updateData as never)
        .eq("id", id);

    if (error) {
        console.error("Failed to update comment", error);
        return { error: error.message };
    }

    revalidatePath("/comments");
    return { success: "Comment updated successfully." };
};

export const deleteCommentAction = async (
    commentId: string,
): Promise<CommentActionState> => {
    if (!commentId) {
        return { error: "Comment ID is required." };
    }

    const supabase = await createSupabaseServerClient();
    const { error } = await supabase.from("comments").delete().eq("id",
commentId);

    if (error) {
        console.error("Failed to delete comment", error);
        return { error: error.message };
    }

    revalidatePath("/comments");
    return { success: "Comment deleted successfully." };
};
</file>

<file path="admin/src/lib/data/orders.ts">
import { createSupabaseServerClient } from "@/lib/supabase/server";
import type { SupabaseOrderRecord } from "@/lib/types";

export type OrdersBoardData = {
  orders: SupabaseOrderRecord[];
};

export const fetchOrdersBoard = async (): Promise<OrdersBoardData> => {
  const supabase = await createSupabaseServerClient();

  const { data, error } = await supabase
    .from("orders")
    .select(

"id,user_id,total_price,total_quantity,status,items,created_at,updated_at",
    )
    .order("created_at", { ascending: false })
    .limit(80);
```

```
  if (error) {
    console.error("Failed to load orders", error);
    return { orders: [] };
  }

  return {
    orders: (data ?? []) as unknown as SupabaseOrderRecord[],
  };
};
</file>

<file path="admin/src/lib/data/photo-bookings-actions.ts">
"use server";

import { revalidatePath } from "next/cache";
import { createSupabaseServerClient } from "@/lib/supabase/server";

export type PhotoBookingActionState = {
    error?: string;
    success?: string;
};

const sanitize = (value: FormDataEntryValue | null) =>
    String(value ?? "").trim();

export const createPhotoBookingAction = async (
    _prev: PhotoBookingActionState,
    formData: FormData,
): Promise<PhotoBookingActionState> => {
    const name = sanitize(formData.get("name"));
    const email = sanitize(formData.get("email"));
    const phone = sanitize(formData.get("phone")) || null;
    const date = sanitize(formData.get("date"));
    const time = sanitize(formData.get("time"));
    const packageType = sanitize(formData.get("package"));
    const people = Number(formData.get("people") ?? 1);
    const message = sanitize(formData.get("message")) || null;

    if (!name || !email || !date || !time || !packageType) {
        return { error: "Name, email, date, time, and package are required." };
    }

    if (Number.isNaN(people) || people <= 0) {
        return { error: "Number of people must be greater than zero." };
    }

    const supabase = await createSupabaseServerClient();
    const bookingData = {
        name,
        email,
        phone,
        date,
        time,
        package: packageType,
        people,
        message,
    };

    const { error } = await supabase
        .from("photo_boot_bookings")
        .insert(bookingData as never);

    if (error) {
```

```
        console.error("Failed to create photo booking", error);
        return { error: error.message };
    }

    revalidatePath("/photo-bookings");
    return { success: "Photo booth booking created successfully." };
};

export const updatePhotoBookingAction = async (
    _prev: PhotoBookingActionState,
    formData: FormData,
): Promise<PhotoBookingActionState> => {
    const id = sanitize(formData.get("id"));
    const name = sanitize(formData.get("name"));
    const email = sanitize(formData.get("email"));
    const phone = sanitize(formData.get("phone")) || null;
    const date = sanitize(formData.get("date"));
    const time = sanitize(formData.get("time"));
    const packageType = sanitize(formData.get("package"));
    const people = Number(formData.get("people") ?? 1);
    const message = sanitize(formData.get("message")) || null;

    if (!id || !name || !email || !date || !time || !packageType) {
        return { error: "All required fields must be provided." };
    }

    const supabase = await createSupabaseServerClient();
    const updateData = {
        name,
        email,
        phone,
        date,
        time,
        package: packageType,
        people,
        message,
    };

    const { error } = await supabase
        .from("photo_boot_bookings")
        .update(updateData as never)
        .eq("id", id);

    if (error) {
        console.error("Failed to update photo booking", error);
        return { error: error.message };
    }

    revalidatePath("/photo-bookings");
    return { success: "Photo booth booking updated successfully." };
};
</file>

<file path="admin/src/lib/data/testimonials-actions.ts">
"use server";

import { revalidatePath } from "next/cache";
import { createSupabaseServerClient } from "@/lib/supabase/server";

export type TestimonialActionState = {
    error?: string;
    success?: string;
};
```

```
const sanitize = (value: FormDataEntryValue | null) =>
    String(value ?? "").trim();

export const createTestimonialAction = async (
    _prev: TestimonialActionState,
    formData: FormData,
): Promise<TestimonialActionState> => {
    const text = sanitize(formData.get("text"));
    const author = sanitize(formData.get("author"));
    const rating = Number(formData.get("rating") ?? 5);

    if (!text || !author) {
        return { error: "Text and author are required." };
    }

    if (Number.isNaN(rating) || rating < 1 || rating > 5) {
        return { error: "Rating must be between 1 and 5." };
    }

    const supabase = await createSupabaseServerClient();
    const testimonialData = {
        text,
        author,
        rating,
    };

    const { error } = await supabase
        .from("testimonials")
        .insert(testimonialData as never);

    if (error) {
        console.error("Failed to create testimonial", error);
        return { error: error.message };
    }

    revalidatePath("/testimonials");
    return { success: "Testimonial created successfully." };
};

export const updateTestimonialAction = async (
    _prev: TestimonialActionState,
    formData: FormData,
): Promise<TestimonialActionState> => {
    const id = sanitize(formData.get("id"));
    const text = sanitize(formData.get("text"));
    const author = sanitize(formData.get("author"));
    const rating = Number(formData.get("rating") ?? 5);

    if (!id || !text || !author) {
        return { error: "ID, text, and author are required." };
    }

    if (Number.isNaN(rating) || rating < 1 || rating > 5) {
        return { error: "Rating must be between 1 and 5." };
    }

    const supabase = await createSupabaseServerClient();
    const updateData = {
        text,
        author,
        rating,
    };

    const { error } = await supabase
```

```
        .from("testimonials")
        .update(updateData as never)
        .eq("id", id);

    if (error) {
        console.error("Failed to update testimonial", error);
        return { error: error.message };
    }

    revalidatePath("/testimonials");
    return { success: "Testimonial updated successfully." };
};

export const deleteTestimonialAction = async (
    testimonialId: string,
): Promise<TestimonialActionState> => {
    if (!testimonialId) {
        return { error: "Testimonial ID is required." };
    }

    const supabase = await createSupabaseServerClient();
    const { error } = await supabase
        .from("testimonials")
        .delete()
        .eq("id", testimonialId);

    if (error) {
        console.error("Failed to delete testimonial", error);
        return { error: error.message };
    }

    revalidatePath("/testimonials");
    return { success: "Testimonial deleted successfully." };
};

export type FeaturedItemActionState = {
    error?: string;
    success?: string;
};

export const createFeaturedItemAction = async (
    _prev: FeaturedItemActionState,
    formData: FormData,
): Promise<FeaturedItemActionState> => {
    const name = sanitize(formData.get("name"));
    const description = sanitize(formData.get("description"));
    const imageUrl = sanitize(formData.get("image_url")) || null;

    if (!name || !description) {
        return { error: "Name and description are required." };
    }

    const supabase = await createSupabaseServerClient();
    const itemData = {
        name,
        description,
        image_url: imageUrl,
    };

    const { error } = await supabase
        .from("featured_items")
        .insert(itemData as never);

    if (error) {
```

```
        console.error("Failed to create featured item", error);
        return { error: error.message };
    }

    revalidatePath("/featured-items");
    return { success: "Featured item created successfully." };
};

export const updateFeaturedItemAction = async (
    _prev: FeaturedItemActionState,
    formData: FormData,
): Promise<FeaturedItemActionState> => {
    const id = sanitize(formData.get("id"));
    const name = sanitize(formData.get("name"));
    const description = sanitize(formData.get("description"));
    const imageUrl = sanitize(formData.get("image_url")) || null;

    if (!id || !name || !description) {
        return { error: "ID, name, and description are required." };
    }

    const supabase = await createSupabaseServerClient();
    const updateData = {
        name,
        description,
        image_url: imageUrl,
    };

    const { error } = await supabase
        .from("featured_items")
        .update(updateData as never)
        .eq("id", id);

    if (error) {
        console.error("Failed to update featured item", error);
        return { error: error.message };
    }

    revalidatePath("/featured-items");
    return { success: "Featured item updated successfully." };
};

export const deleteFeaturedItemAction = async (
    itemId: string,
): Promise<FeaturedItemActionState> => {
    if (!itemId) {
        return { error: "Item ID is required." };
    }

    const supabase = await createSupabaseServerClient();
    const { error } = await supabase.from("featured_items").delete().eq("id",
itemId);

    if (error) {
        console.error("Failed to delete featured item", error);
        return { error: error.message };
    }

    revalidatePath("/featured-items");
    return { success: "Featured item deleted successfully." };
};
</file>

<file path="admin/src/lib/supabase/config.ts">
```

```
export const getSupabaseConfig = () => {
  const url = process.env.NEXT_PUBLIC_SUPABASE_URL;
  const anonKey = process.env.NEXT_PUBLIC_SUPABASE_PUBLISHABLE_KEY;

  if (!url) {
    throw new Error(
      "Missing NEXT_PUBLIC_SUPABASE_URL environment variable. Please add it to
your .env.local file."
    );
  }

  if (!anonKey) {
    throw new Error(
      "Missing NEXT_PUBLIC_SUPABASE_PUBLISHABLE_KEY environment variable. Please
add it to your .env.local file."
    );
  }

  return { url, anonKey };
};
</file>

<file path="admin/src/lib/supabase/server.ts">
import { createServerClient } from "@supabase/ssr";
import { cookies } from "next/headers";
import { getSupabaseConfig } from "./config";
import { Database } from "@/lib/database.types";

export const createSupabaseServerClient = async () => {
  const cookieStore = await cookies();
  const { url, anonKey } = getSupabaseConfig();

  return createServerClient<Database>(url, anonKey, {
    cookies: {
      getAll() {
        return cookieStore.getAll();
      },
      setAll(cookiesToSet) {
        try {
          cookiesToSet.forEach(({ name, value }) => {
            cookieStore.set(name, value);
          });
        } catch {
          // noop: called from a Server Component
        }
      },
    },
  });
};

export type SupabaseServerClient = Awaited<
  ReturnType<typeof createSupabaseServerClient>
>;
</file>

<file path="admin/src/lib/types/orders.ts">
import type { PaymentMethod } from "./base";
import type { Product } from "./products";

export interface OrderItem {
  name: string;
  quantity: number;
}
```

```
export type CartItem = Product & {
  quantity: number;
};

export interface OrderDetails {
  userId: string;
  items: CartItem[];
  totalAmount: number;
  shippingAddress: string;
  paymentMethod: PaymentMethod;
}

export type OrderStatus = "pending" | "processing" | "completed" | "cancelled" |
"deleted";
export type OrderType = "Takeaway" | "Photo boot" | "Laundry";

export interface Order {
  orderId: string;
  userId?: string;
  orderDate: Date;
  products: CartItem[];
  quantity: number;
  totalAmount: number;
  status: OrderStatus;
  type: OrderType;
  location?: string;
  paymentMethod?: PaymentMethod;
  deliveryAddress?: string;
  shippingAddress?: string;
}

export interface SupabaseOrderRecord {
  id: string;
  user_id: string | null;
  items: CartItem[];
  total_price: number;
  total_quantity: number;
  status: OrderStatus;
  created_at: string;
  updated_at?: string;
}
</file>

<file path="client/src/app/(auth)/confirm-email/page.tsx">
import React from 'react';
import Main from '@/components/ui/layout/Main';
import React from 'react';
import Main from '@/components/ui/layout/Main';
import { HiEnvelope } from 'react-icons/hi2';
import AppLink from '@/components/ui/Link';

export default function ConfirmEmailPage() {
    return (
        <Main
            tittle='Confirm Email'
            Icon={HiEnvelope}
            className='flex items-center justify-center min-h-[60vh] p-4
md:p-8'>
            <div className='max-w-md w-full bg-gradient-to-br from-yellow-900/90
to-yellow-800/90 backdrop-blur-xl p-6 md:p-10 rounded-3xl border border-white/10
shadow-2xl text-center transform transition-all hover:scale-[1.01]
duration-300'>
                <div className='bg-gradient-to-br from-amber-500/20 to-
yellow-500/20 p-5 md:p-6 rounded-full border border-amber-400/30 inline-block
```

```
mb-6 shadow-lg shadow-amber-500/10 animate-pulse'>
                    <HiEnvelope className='text-5xl md:text-6xl text-
amber-400' />
                </div>
                <h2 className='text-2xl md:text-3xl font-bold text-white mb-4
tracking-tight'>Check your email</h2>
                <p className='text-yellow-100/90 mb-8 text-sm md:text-base
leading-relaxed'>
                    We've sent you a confirmation link. Please check your inbox
(and spam folder) to verify your account.
                </p>
                <div className='space-y-4'>
                    <AppLink href='/login' variant='button' className='w-full
py-3 text-lg shadow-amber-500/20'>
                        Back to Login
                    </AppLink>
                    <p className='text-xs text-yellow-500/60 mt-4'>
                        Didn't receive the email? Check your spam folder or try
logging in again.
                    </p>
                </div>
            </div>
        </Main>
    );
}
</file>

<file path=".repomix/bundles.json">
{
  "bundles": {
    "context-532": {
      "name": "context",
      "created": "2025-11-27T16:24:56.957Z",
      "lastUsed": "2025-11-29T11:20:56.319Z",
      "tags": [],
      "files": [
        "admin",
        "client"
      ]
    }
  }
}
</file>

<file path="admin/src/components/auth/AuthShell.tsx">
import type { ReactNode } from "react";

type Props = {
  title: string;
  description: string;
  children: ReactNode;
  footer?: ReactNode;
};

export const AuthShell = ({ title, description, children, footer }: Props) => (
  <div className="w-full max-w-md rounded-2xl border border-white/10 bg-
yellow-900/80 p-8 text-white shadow-2xl backdrop-blur">
    <div className="space-y-2 text-center">
      <h1 className="text-2xl font-semibold">{title}</h1>
      <p className="text-sm text-yellow-300">{description}</p>
    </div>

    <div className="mt-6">{children}</div>
```

```
    {footer ? (
      <div className="mt-6 text-center text-sm text-yellow-400">{footer}</div>
    ) : null}
  </div>
);
</file>

<file path="admin/src/components/dashboard/LowInventoryCard.tsx">
import { HiOutlineExclamationTriangle } from "react-icons/hi2";
import { DashboardShell } from "@/components/layout/DashboardShell";
import type { ProductRecord } from "@/lib/types";

type Props = {
  items: ProductRecord[];
};

export const LowInventoryCard = ({ items }: Props) => (
  <DashboardShell
    title="Low inventory"
    description="Keep signature menu items available during peak demand."
  >
    {items.length === 0 ? (
      <p className="text-sm text-yellow-500">
        All tracked items are within healthy stock levels.
      </p>
    ) : (
      <ul className="space-y-3">
        {items.map((item) => (
          <li
            key={item.id}
            className="flex items-center justify-between rounded-xl border
border-rose-500/30 bg-rose-500/5 px-3 py-2 text-sm text-rose-200"
          >
            <div>
              <p className="font-semibold text-white">{item.name}</p>
              <p className="text-xs uppercase tracking-[0.3em] text-rose-300">
                {item.category_name ?? "Uncategorised"}
              </p>
            </div>
            <div className="flex items-center gap-1 text-rose-300">
              <HiOutlineExclamationTriangle />
              <span>{item.stock ?? 0} left</span>
            </div>
          </li>
        ))}
      </ul>
    )}
  </DashboardShell>
);
</file>

<file path="admin/src/components/featured-items/FeaturedItemsBoard.tsx">
"use client";

import { useState } from "react";
import { useFormState } from "react-dom";
import { HiOutlinePlus } from "react-icons/hi2";
import { SubmitButton } from "@/components/forms/SubmitButton";
import { DeleteConfirmDialog } from "@/components/shared/DeleteConfirmDialog";
import {
    type FeaturedItemActionState,
    createFeaturedItemAction,
    updateFeaturedItemAction,
    deleteFeaturedItemAction,
```

```
} from "@/lib/data/testimonials-actions";
import type { FeaturedItemRecord } from "@/lib/types";
import { formatDistanceToNow } from "date-fns";

type Props = {
    items: FeaturedItemRecord[];
};

const initialState: FeaturedItemActionState = {};

export const FeaturedItemsBoard = ({ items }: Props) => {
    const [search, setSearch] = useState("");
    const [editingId, setEditingId] = useState<string | null>(null);
    const [deleteId, setDeleteId] = useState<string | null>(null);
    const [showCreateForm, setShowCreateForm] = useState(false);

    const filtered = items.filter(
        (item) =>
            !search ||
            item.name.toLowerCase().includes(search.toLowerCase()) ||
            item.description.toLowerCase().includes(search.toLowerCase()),
    );

    const handleDelete = async () => {
        if (deleteId) {
            await deleteFeaturedItemAction(deleteId);
            setDeleteId(null);
        }
    };

    return (
        <div className="space-y-4">
            <div className="flex items-center justify-between gap-3">
                <input
                    type="search"
                    placeholder="Search featured items..."
                    value={search}
                    onChange={(e) => setSearch(e.target.value)}
                    className="w-full max-w-md rounded-full border border-
white/10 bg-yellow-900/60 px-4 py-3 text-white outline-none focus:border-
indigo-400 focus:ring-2 focus:ring-indigo-400/40"
                />
                <button
                    type="button"
                    onClick={() => setShowCreateForm(true)}
                    className="flex items-center gap-2 rounded-full bg-
indigo-600 px-4 py-3 text-sm font-medium text-white hover:bg-indigo-700"
                >
                    <HiOutlinePlus className="h-5 w-5" />
                    <span className="whitespace-nowrap">Add Featured Item</span>
                </button>
            </div>

            {showCreateForm && (
                <CreateFeaturedItemForm onCancel={() =>
setShowCreateForm(false)} />
            )}

            <div className="grid gap-4 md:grid-cols-2 lg:grid-cols-3">
                {filtered.map((item) => (
                    <FeaturedItemCard
                        key={item.id}
                        item={item}
                        isEditing={editingId === item.id}
```

```
                        onEdit={() => setEditingId(item.id)}
                        onCancelEdit={() => setEditingId(null)}
                        onDelete={() => setDeleteId(item.id)}
                    />
                ))}
            </div>

            <DeleteConfirmDialog
                isOpen={deleteId !== null}
                onClose={() => setDeleteId(null)}
                onConfirm={handleDelete}
                title="Delete Featured Item"
                message="Are you sure you want to delete this featured item?
This action cannot be undone."
            />
        </div>
    );
};

type CreateFeaturedItemFormProps = {
    onCancel: () => void;
};

const CreateFeaturedItemForm = ({ onCancel }: CreateFeaturedItemFormProps) => {
    const [state, formAction] = useFormState(createFeaturedItemAction,
initialState);

    return (
        <div className="rounded-2xl border border-indigo-500/30 bg-indigo-500/5
p-4">
            <h3 className="mb-3 text-sm font-semibold text-white">Create New
Featured Item</h3>
            <form action={formAction} className="space-y-3">
                <label className="block space-y-2 text-sm">
                    <span className="text-yellow-300">Name *</span>
                    <input
                        name="name"
                        required
                        className="w-full rounded-lg border border-white/10 bg-
yellow-900/60 px-3 py-2 text-white outline-none focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
                    />
                </label>

                <label className="block space-y-2 text-sm">
                    <span className="text-yellow-300">Image URL</span>
                    <input
                        name="image_url"
                        type="url"
                        placeholder="https://example.com/image.jpg"
                        className="w-full rounded-lg border border-white/10 bg-
yellow-900/60 px-3 py-2 text-white outline-none focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
                    />
                </label>

                <label className="block space-y-2 text-sm">
                    <span className="text-yellow-300">Description *</span>
                    <textarea
                        name="description"
                        rows={4}
                        required
                        className="w-full rounded-lg border border-white/10 bg-
yellow-900/60 px-3 py-2 text-white outline-none focus:border-indigo-400
```

```
focus:ring-2 focus:ring-indigo-400/40"
                    />
                </label>

                {state.error && <p className="text-sm text-
rose-400">{state.error}</p>}
                {state.success && (
                    <p className="text-sm text-emerald-400">{state.success}</p>
                )}

                <div className="flex gap-2">
                    <SubmitButton
                        label="Create Featured Item"
                        loadingLabel="Creating..."
                        className="bg-indigo-600 px-4 py-2 text-sm"
                    />
                    <button
                        type="button"
                        onClick={onCancel}
                        className="rounded-lg bg-yellow-800 px-4 py-2 text-sm
text-white hover:bg-yellow-700"
                    >
                        Cancel
                    </button>
                </div>
            </form>
        </div>
    );
};

type FeaturedItemCardProps = {
    item: FeaturedItemRecord;
    isEditing: boolean;
    onEdit: () => void;
    onCancelEdit: () => void;
    onDelete: () => void;
};

const FeaturedItemCard = ({
    item,
    isEditing,
    onEdit,
    onCancelEdit,
    onDelete,
}: FeaturedItemCardProps) => {
    const [state, formAction] = useFormState(
        updateFeaturedItemAction,
        initialState,
    );

    return (
        <div className="rounded-2xl border border-white/10 bg-yellow-900/40
p-4">
            {isEditing ? (
                <form action={formAction} className="space-y-3">
                    <input type="hidden" name="id" value={item.id} />

                    <label className="block space-y-2 text-sm">
                        <span className="text-yellow-300">Name</span>
                        <input
                            name="name"
                            defaultValue={item.name}
                            required
                            className="w-full rounded-lg border border-white/10
```

```
bg-yellow-900/60 px-3 py-2 text-white outline-none focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
                        />
                    </label>

                    <label className="block space-y-2 text-sm">
                        <span className="text-yellow-300">Image URL</span>
                        <input
                            name="image_url"
                            type="url"
                            defaultValue={item.image_url || ""}
                            placeholder="https://example.com/image.jpg"
                            className="w-full rounded-lg border border-white/10
bg-yellow-900/60 px-3 py-2 text-white outline-none focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
                        />
                    </label>

                    <label className="block space-y-2 text-sm">
                        <span className="text-yellow-300">Description</span>
                        <textarea
                            name="description"
                            rows={4}
                            defaultValue={item.description}
                            required
                            className="w-full rounded-lg border border-white/10
bg-yellow-900/60 px-3 py-2 text-white outline-none focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
                        />
                    </label>

                    {state.error && <p className="text-sm text-
rose-400">{state.error}</p>}
                    {state.success && (
                        <p className="text-sm text-emerald-400">{state.success}
</p>
                    )}

                    <div className="flex gap-2">
                        <SubmitButton
                            label="Save"
                            loadingLabel="Saving..."
                            className="bg-indigo-600 px-4 py-2 text-sm"
                        />
                        <button
                            type="button"
                            onClick={onCancelEdit}
                            className="rounded-lg bg-yellow-800 px-4 py-2 text-
sm text-white hover:bg-yellow-700"
                        >
                            Cancel
                        </button>
                    </div>
                </form>
            ) : (
                <>
                    {item.image_url && (
                        <div className="mb-3">
                            <img
                                src={item.image_url}
                                alt={item.name}
                                className="h-48 w-full rounded-lg object-cover"
                            />
                        </div>
```

```
                    )}

                    <div className="flex items-start justify-between">
                        <h3 className="flex-1 text-lg font-semibold text-
white">{item.name}</h3>
                        <div className="flex gap-2">
                            <button
                                type="button"
                                onClick={onEdit}
                                className="rounded-lg bg-yellow-800 px-3 py-1
text-xs text-white hover:bg-yellow-700"
                            >
                                Edit
                            </button>
                            <button
                                type="button"
                                onClick={onDelete}
                                className="rounded-lg bg-rose-600 px-3 py-1
text-xs text-white hover:bg-rose-700"
                            >
                                Delete
                            </button>
                        </div>
                    </div>

                    <p className="mt-3 text-sm text-
yellow-300">{item.description}</p>

                    <div className="mt-3 flex justify-between text-xs text-
yellow-500">
                        <div className="flex gap-4">
                            <span>{item.likes.length} likes</span>
                            <span>{Array.isArray(item.comments) ?
item.comments.length : 0} comments</span>
                        </div>
                        <span>
                            {formatDistanceToNow(new Date(item.created_at), {
                                addSuffix: true,
                            })}
                        </span>
                    </div>
                </>
            )}
        </div>
    );
};
</file>

<file path="admin/src/components/orders/UpdateOrderStatusForm.tsx">
"use client";

import { useFormState } from "react-dom";
import { SubmitButton } from "@/components/forms/SubmitButton";
import {
  type OrderActionState,
  updateOrderStatusAction,
} from "@/lib/data/orders-actions";
import type { OrderStatus } from "@/lib/types";

type Props = {
  orderId: string;
  currentStatus: OrderStatus;
};
```

```
const initialState: OrderActionState = {};
const STATUSES: OrderStatus[] = [
  "pending",
  "processing",
  "completed",
  "cancelled",
  "deleted",
];

export const UpdateOrderStatusForm = ({ orderId, currentStatus }: Props) => {
  const [state, formAction] = useFormState(
    updateOrderStatusAction,
    initialState,
  );

  return (
    <form action={formAction} className="space-y-2 text-sm text-yellow-200">
      <input type="hidden" name="orderId" value={orderId} />
      <label className="space-y-1 text-xs uppercase tracking-[0.3em] text-
yellow-500">
        Status
        <select
          name="status"
          defaultValue={currentStatus}
          className="mt-1 w-full rounded-lg border border-white/10 bg-
yellow-900/60 px-3 py-2 text-white outline-none focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
        >
          {STATUSES.map((status) => (
            <option key={status} value={status}>
              {status}
            </option>
          ))}
        </select>
      </label>

      {state.error ? (
        <p className="text-xs text-rose-400">{state.error}</p>
      ) : null}
      {state.success ? (
        <p className="text-xs text-emerald-400">{state.success}</p>
      ) : null}

      <SubmitButton
        label="Update status"
        loadingLabel="Updating..."
        className="bg-yellow-800 px-3 py-2 text-xs"
      />
    </form>
  );
};
</file>

<file path="admin/src/components/testimonials/TestimonialsBoard.tsx">
"use client";

import { useState } from "react";
import { useFormState } from "react-dom";
import { HiOutlinePlus } from "react-icons/hi2";
import { SubmitButton } from "@/components/forms/SubmitButton";
import { DeleteConfirmDialog } from "@/components/shared/DeleteConfirmDialog";
import {
    type TestimonialActionState,
    createTestimonialAction,
```

```
    updateTestimonialAction,
    deleteTestimonialAction,
} from "@/lib/data/testimonials-actions";
import type { TestimonialRecord } from "@/lib/types";
import { formatDistanceToNow } from "date-fns";

type Props = {
    testimonials: TestimonialRecord[];
};

const initialState: TestimonialActionState = {};

export const TestimonialsBoard = ({ testimonials }: Props) => {
    const [search, setSearch] = useState("");
    const [editingId, setEditingId] = useState<string | null>(null);
    const [deleteId, setDeleteId] = useState<string | null>(null);
    const [showCreateForm, setShowCreateForm] = useState(false);

    const filtered = testimonials.filter(
        (testimonial) =>
            !search ||
            testimonial.text.toLowerCase().includes(search.toLowerCase()) ||
            testimonial.author.toLowerCase().includes(search.toLowerCase()),
    );

    const handleDelete = async () => {
        if (deleteId) {
            await deleteTestimonialAction(deleteId);
            setDeleteId(null);
        }
    };

    return (
        <div className="space-y-4">
            <div className="flex items-center justify-between gap-3">
                <input
                    type="search"
                    placeholder="Search testimonials..."
                    value={search}
                    onChange={(e) => setSearch(e.target.value)}
                    className="w-full max-w-md rounded-full border border-
white/10 bg-yellow-900/60 px-4 py-3 text-white outline-none focus:border-
indigo-400 focus:ring-2 focus:ring-indigo-400/40"
                />
                <button
                    type="button"
                    onClick={() => setShowCreateForm(true)}
                    className="flex items-center gap-2 rounded-full bg-
indigo-600 px-4 py-3 text-sm font-medium text-white hover:bg-indigo-700"
                >
                    <HiOutlinePlus className="h-5 w-5" />
                    <span className="whitespace-nowrap">Add Testimonial</span>
                </button>
            </div>

            {showCreateForm && (
                <CreateTestimonialForm onCancel={() => setShowCreateForm(false)}
/>
            )}

            <div className="grid gap-4 md:grid-cols-2">
                {filtered.map((testimonial) => (
                    <TestimonialCard
                        key={testimonial.id}
```

```
                        testimonial={testimonial}
                        isEditing={editingId === testimonial.id}
                        onEdit={() => setEditingId(testimonial.id)}
                        onCancelEdit={() => setEditingId(null)}
                        onDelete={() => setDeleteId(testimonial.id)}
                    />
                ))}
            </div>

            <DeleteConfirmDialog
                isOpen={deleteId !== null}
                onClose={() => setDeleteId(null)}
                onConfirm={handleDelete}
                title="Delete Testimonial"
                message="Are you sure you want to delete this testimonial? This
action cannot be undone."
            />
        </div>
    );
};

type CreateTestimonialFormProps = {
    onCancel: () => void;
};

const CreateTestimonialForm = ({ onCancel }: CreateTestimonialFormProps) => {
    const [state, formAction] = useFormState(createTestimonialAction,
initialState);

    return (
        <div className="rounded-2xl border border-indigo-500/30 bg-indigo-500/5
p-4">
            <h3 className="mb-3 text-sm font-semibold text-white">Create New
Testimonial</h3>
            <form action={formAction} className="space-y-3">
                <label className="block space-y-2 text-sm">
                    <span className="text-yellow-300">Author *</span>
                    <input
                        name="author"
                        required
                        className="w-full rounded-lg border border-white/10 bg-
yellow-900/60 px-3 py-2 text-white outline-none focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
                    />
                </label>

                <label className="block space-y-2 text-sm">
                    <span className="text-yellow-300">Rating</span>
                    <select
                        name="rating"
                        defaultValue="5"
                        className="w-full rounded-lg border border-white/10 bg-
yellow-900/60 px-3 py-2 text-white outline-none focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
                    >
                        <option value="1">1 Star</option>
                        <option value="2">2 Stars</option>
                        <option value="3">3 Stars</option>
                        <option value="4">4 Stars</option>
                        <option value="5">5 Stars</option>
                    </select>
                </label>

                <label className="block space-y-2 text-sm">
```

```
                    <span className="text-yellow-300">Testimonial Text *</span>
                    <textarea
                        name="text"
                        rows={4}
                        required
                        className="w-full rounded-lg border border-white/10 bg-
yellow-900/60 px-3 py-2 text-white outline-none focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
                    />
                </label>

                {state.error && <p className="text-sm text-
rose-400">{state.error}</p>}
                {state.success && (
                    <p className="text-sm text-emerald-400">{state.success}</p>
                )}

                <div className="flex gap-2">
                    <SubmitButton
                        label="Create Testimonial"
                        loadingLabel="Creating..."
                        className="bg-indigo-600 px-4 py-2 text-sm"
                    />
                    <button
                        type="button"
                        onClick={onCancel}
                        className="rounded-lg bg-yellow-800 px-4 py-2 text-sm
text-white hover:bg-yellow-700"
                    >
                        Cancel
                    </button>
                </div>
            </form>
        </div>
    );
};

type TestimonialCardProps = {
    testimonial: TestimonialRecord;
    isEditing: boolean;
    onEdit: () => void;
    onCancelEdit: () => void;
    onDelete: () => void;
};

const TestimonialCard = ({
    testimonial,
    isEditing,
    onEdit,
    onCancelEdit,
    onDelete,
}: TestimonialCardProps) => {
    const [state, formAction] = useFormState(
        updateTestimonialAction,
        initialState,
    );

    return (
        <div className="rounded-2xl border border-white/10 bg-yellow-900/40
p-4">
            {isEditing ? (
                <form action={formAction} className="space-y-3">
                    <input type="hidden" name="id" value={testimonial.id} />
```

```
                    <label className="block space-y-2 text-sm">
                        <span className="text-yellow-300">Author</span>
                        <input
                            name="author"
                            defaultValue={testimonial.author}
                            required
                            className="w-full rounded-lg border border-white/10
bg-yellow-900/60 px-3 py-2 text-white outline-none focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
                        />
                    </label>

                    <label className="block space-y-2 text-sm">
                        <span className="text-yellow-300">Rating</span>
                        <select
                            name="rating"
                            defaultValue={testimonial.rating}
                            className="w-full rounded-lg border border-white/10
bg-yellow-900/60 px-3 py-2 text-white outline-none focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
                        >
                            <option value="1">1 Star</option>
                            <option value="2">2 Stars</option>
                            <option value="3">3 Stars</option>
                            <option value="4">4 Stars</option>
                            <option value="5">5 Stars</option>
                        </select>
                    </label>

                    <label className="block space-y-2 text-sm">
                        <span className="text-yellow-300">Testimonial
Text</span>
                        <textarea
                            name="text"
                            rows={4}
                            defaultValue={testimonial.text}
                            required
                            className="w-full rounded-lg border border-white/10
bg-yellow-900/60 px-3 py-2 text-white outline-none focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
                        />
                    </label>

                    {state.error && <p className="text-sm text-
rose-400">{state.error}</p>}
                    {state.success && (
                        <p className="text-sm text-emerald-400">{state.success}
</p>
                    )}

                    <div className="flex gap-2">
                        <SubmitButton
                            label="Save"
                            loadingLabel="Saving..."
                            className="bg-indigo-600 px-4 py-2 text-sm"
                        />
                        <button
                            type="button"
                            onClick={onCancelEdit}
                            className="rounded-lg bg-yellow-800 px-4 py-2 text-
sm text-white hover:bg-yellow-700"
                        >
                            Cancel
                        </button>
```

```
                    </div>
                </form>
            ) : (
                <>
                    <div className="flex items-start justify-between">
                        <div className="flex-1">
                            <div className="flex items-center gap-2">
                                <h3 className="font-semibold text-
white">{testimonial.author}</h3>
                                <div className="flex gap-0.5">
                                    {Array.from({ length:
testimonial.rating }).map((_, i) => (
                                        <span key={i} className="text-
yellow-400">â˜…</span>
                                    ))}
                                </div>
                            </div>
                            <p className="mt-1 text-xs text-yellow-500">
                                {formatDistanceToNow(new
Date(testimonial.created_at), {
                                    addSuffix: true,
                                })}
                            </p>
                        </div>
                        <div className="flex gap-2">
                            <button
                                type="button"
                                onClick={onEdit}
                                className="rounded-lg bg-yellow-800 px-3 py-1
text-xs text-white hover:bg-yellow-700"
                            >
                                Edit
                            </button>
                            <button
                                type="button"
                                onClick={onDelete}
                                className="rounded-lg bg-rose-600 px-3 py-1
text-xs text-white hover:bg-rose-700"
                            >
                                Delete
                            </button>
                        </div>
                    </div>

                    <p className="mt-3 text-sm text-
yellow-300">{testimonial.text}</p>

                    <div className="mt-3 flex gap-4 text-xs text-yellow-500">
                        <span>{testimonial.likes.length} likes</span>
                        <span>{Array.isArray(testimonial.comments) ?
testimonial.comments.length : 0} comments</span>
                    </div>
                </>
            )}
        </div>
    );
};
</file>

<file path="admin/src/lib/data/products.ts">
import { createSupabaseServerClient } from "@/lib/supabase/server";
import type { ProductRecord } from "@/lib/types";

export const fetchProductCatalog = async (): Promise<ProductRecord[]> => {
```

```
  const supabase = await createSupabaseServerClient();
  const { data, error } = await supabase
    .from("products")
    .select(

"id,name,category_name,price,stock,badge,description,image_url,created_at",
    )
    .order("created_at", { ascending: false });

  if (error) {
    console.error("Failed to fetch products", error);
    return [];
  }

  return (data ?? []) as ProductRecord[];
};
</file>

<file path="admin/src/lib/data/profile-details.ts">
import { createSupabaseServerClient } from "@/lib/supabase/server";
import type { SupabaseOrderRecord } from "@/lib/types";

export const fetchProfileOrders = async (profileId: string):
Promise<SupabaseOrderRecord[]> => {
    const supabase = await createSupabaseServerClient();
    const { data, error } = await supabase
        .from("orders")
        .select("*")
        .eq("user_id", profileId)
        .order("created_at", { ascending: false });

    if (error) {
        console.error("Failed to fetch profile orders", error);
        return [];
    }

    return (data ?? []) as unknown as SupabaseOrderRecord[];
};

export const fetchProfileFavorites = async (profileId: string) => {
    const supabase = await createSupabaseServerClient();
    const { data, error } = await supabase
        .from("user_favorites")
        .select(`
      *,
      products (
        id,
        name,
        image_url,
        price
      )
    `)
        .eq("user_id", profileId);

    if (error) {
        console.error("Failed to fetch profile favorites", error);
        return [];
    }

    return data ?? [];
};
</file>

<file path="admin/src/lib/data/profiles-actions.ts">
```

```
"use server";

import { revalidatePath } from "next/cache";
import { createSupabaseServerClient } from "@/lib/supabase/server";

export type ProfileActionState = {
    error?: string;
    success?: string;
};

const sanitize = (value: FormDataEntryValue | null) =>
    String(value ?? "").trim();

export const updateProfileAction = async (
    _prev: ProfileActionState,
    formData: FormData,
): Promise<ProfileActionState> => {
    const id = sanitize(formData.get("id"));
    const displayName = sanitize(formData.get("display_name")) || null;
    const phone = sanitize(formData.get("phone")) || null;
    const address = sanitize(formData.get("address")) || null;
    const city = sanitize(formData.get("city")) || null;
    const state = sanitize(formData.get("state")) || null;
    const zipCode = sanitize(formData.get("zip_code")) || null;
    const country = sanitize(formData.get("country")) || null;
    const role = sanitize(formData.get("role")) || "customer";

    if (!id) {
        return { error: "Profile ID is required." };
    }

    const supabase = await createSupabaseServerClient();
    const updateData = {
        display_name: displayName,
        phone,
        address,
        city,
        state,
        zip_code: zipCode,
        country,
        role,
        updated_at: new Date().toISOString(),
    };

    const { error } = await supabase
        .from("profiles")
        .update(updateData as never)
        .eq("id", id);

    if (error) {
        console.error("Failed to update profile", error);
        return { error: error.message };
    }

    // Sync with admins table
    if (role === 'admin') {
        const { error: adminError } = await supabase
            .from('admins')
            .upsert({ user_id: id }, { onConflict: 'user_id' });

        if (adminError) {
            console.error("Failed to add to admins table", adminError);
            // Optional: revert profile update or return warning
        }
```

```
    } else {
        const { error: adminError } = await supabase
            .from('admins')
            .delete()
            .eq('user_id', id);

        if (adminError) {
            console.error("Failed to remove from admins table", adminError);
        }
    }

    revalidatePath("/profiles");
    return { success: "Profile updated successfully." };
};
</file>

<file path="admin/src/lib/supabase/storage.ts">
import { createSupabaseBrowserClient } from "./client";

export const STORAGE_BUCKET = "pa-luxe-creation";

export const uploadImage = async (file: File, path: string) => {
    const supabase = createSupabaseBrowserClient();
    const fileExt = file.name.split(".").pop();
    const fileName = `${Math.random().toString(36).substring(2)}.${fileExt}`;
    const filePath = `${path}/${fileName}`;

    console.log("Uploading file to path:", filePath);
    const { error: uploadError } = await supabase.storage
        .from(STORAGE_BUCKET)
        .upload(filePath, file);

    if (uploadError) {
        console.error("Upload error:", uploadError);
        throw uploadError;
    }

    const { data } =
supabase.storage.from(STORAGE_BUCKET).getPublicUrl(filePath);
    console.log("Uploaded file public URL:", data.publicUrl);

    return data.publicUrl;
};

export const deleteImage = async (urlOrPath: string) => {
    console.log("Deleting image:", urlOrPath);
    const supabase = createSupabaseBrowserClient();

    // Extract path from URL if a full URL was provided
    let filePath = urlOrPath;
    if (urlOrPath.includes(STORAGE_BUCKET)) {
        const parts = urlOrPath.split(`${STORAGE_BUCKET}/`);
        filePath = parts[parts.length - 1];
    }
    console.log("Parsed file path for deletion:", filePath);

    const { error } = await supabase.storage
        .from(STORAGE_BUCKET)
        .remove([filePath]);

    if (error) {
        console.error("Failed to delete image:", error);
        throw error;
    }
```

```
    console.log("Image deleted successfully");
};
</file>

<file path="admin/src/lib/types/index.ts">
export * from "./base";
export * from "./orders";
export * from "./products";
export * from "./users";
export * from "./database-records";
export * from "./testimonials";
</file>

<file path="client/.gitignore">
# See https://help.github.com/articles/ignoring-files/ for more about ignoring
files.

# dependencies
/node_modules
/.pnp
.pnp.*
.yarn/*
!.yarn/patches
!.yarn/plugins
!.yarn/releases
!.yarn/versions

# testing
/coverage

# next.js
/.next/
/out/

# production
/build

# misc
.DS_Store
*.pem

# debug
npm-debug.log*
yarn-debug.log*
yarn-error.log*
.pnpm-debug.log*

# env files (can opt-in for committing if needed)
.env*

# vercel
.vercel

# typescript
*.tsbuildinfo
next-env.d.ts
</file>

<file path="client/biome.json">
{
  "$schema": "https://biomejs.dev/schemas/2.2.0/schema.json",
  "vcs": {
    "enabled": true,
    "clientKind": "git",
```

```
    "useIgnoreFile": true
  },
  "files": {
    "ignoreUnknown": true,
    "includes": ["**", "!node_modules", "!.next", "!dist", "!build"]
  },
  "formatter": {
    "enabled": true,
    "indentStyle": "space",
    "indentWidth": 2
  },
  "linter": {
    "enabled": true,
    "rules": {
      "recommended": true,
      "suspicious": {
        "noUnknownAtRules": "off"
      }
    },
    "domains": {
      "next": "recommended",
      "react": "recommended"
    }
  },
  "assist": {
    "actions": {
      "source": {
        "organizeImports": "on"
      }
    }
  }
}
</file>

<file path="client/postcss.config.mjs">
const config = {
  plugins: {
    "@tailwindcss/postcss": {},
  },
};

export default config;
</file>

<file path="client/public/file.svg">
<svg fill="none" viewBox="0 0 16 16" xmlns="http://www.w3.org/2000/svg"><path
d="M14.5 13.5V5.41a1 1 0 0 0-.3-.7L9.8.29A1 1 0 0 0 9.08 0H1.5v13.5A2.5 2.5 0 0
0 4 16h8a2.5 2.5 0 0 0 2.5-2.5m-1.5 0v-7H8v-5H3v12a1 1 0 0 0 1 1h8a1 1 0 0 0 1-
1M9.5 5V2.12L12.38 5zM5.13 5h-.62v1.25h2.12V5zm-.62 3h7.12v1.25H4.5zm.62
3h-.62v1.25h7.12V11z" clip-rule="evenodd" fill="#666"
fill-rule="evenodd"/></svg>
</file>

<file path="client/public/globe.svg">
<svg fill="none" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 16 16"><g clip-
path="url(#a)"><path fill-rule="evenodd" clip-rule="evenodd" d="M10.27 14.1a6.5
6.5 0 0 0 3.67-3.45q-1.24.21-2.7.34-.31 1.83-.97 3.1M8 16A8 8 0 1 0 8 0a8 8 0 0
0 0 16m.48-1.52a7 7 0 0 1-.96 0H7.5a4 4 0 0 1-.84-1.32q-.38-.89-.63-2.08a40 40 0
0 0 3.92 0q-.25 1.2-.63 2.08a4 4 0 0 1-.84 1.31zm2.94-4.76q1.66-.15 2.95-.43a7 7
0 0 0 0-2.58q-1.3-.27-2.95-.43a18 18 0 0 1 0 3.44m-1.27-3.54a17 17 0 0 1 0 3.64
39 39 0 0 1-4.3 0 17 17 0 0 1 0-3.64 39 39 0 0 1 4.3 0m1.1-1.17q1.45.13
2.69.34a6.5 6.5 0 0 0-3.67-3.44q.65 1.26.98 3.1M8.48 1.5l.01.02q.41.37.84
1.31.38.89.63 2.08a40 40 0 0 0-3.92 0q.25-1.2.63-2.08a4 4 0 0 1 .85-1.32 7 7 0 0
1 .96 0m-2.75.4a6.5 6.5 0 0 0-3.67 3.44 29 29 0 0 1 2.7-.34q.31-1.83.97-3.1M4.58
```

6.28q-1.66.16-2.95.43a7 7 0 0 0 0 2.58q1.3.27 2.95.43a18 18 0 0 1 0-3.44m.17 4.71q-1.45-.12-2.69-.34a6.5 6.5 0 0 0 0 3.67 3.44q-.65-1.27-.98-3.1" fill="#666"/></g><defs><clipPath id="a"><path fill="#fff" d="M0 0h16v16H0z"/></clipPath></defs></svg>
</file>

<file path="client/public/next.svg">
<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 394 80"><path fill="#000" d="M262 0h68.5v12.7h-27.2v66.6h-13.6V12.7H262V0ZM149 0v12.7H94v20.4h44.3v12.6H94v21h55v12.6H80.5V0h68.7zm34.3 0h-17.8l63.8 79.4h17.9l-32-39.7 32-39.6h-17.9l-23 28.6-23-28.6zm18.3 56.7-9-11-27.1 33.7h17.8l18.3-22.7z"/><path fill="#000" d="M81 79.3 17 0H0v79.3h13.6V17l50.2 62.3H81Zm252.6-.4c-1 0-1.8-.4-2.5-1s-1.1-1.6-1.1-2.6.3-1.8 1-2.5 1.6-1 2.6-1 1.8.3 2.5 1a3.4 3.4 0 0 1 .6 4.3 3.7 3.7 0 0 1-3 1.8zm23.2-33.5h6v23.3c0 2.1-.4 4-1.3 5.5a9.1 9.1 0 0 1-3.8 3.5c-1.6.8-3.5 1.3-5.7 1.3-2 0-3.7-.4-5.3-1s-2.8-1.8-3.7-3.2c-.9-1.3-1.4-3-1.4-5h6c.1.8.3 1.6.7 2.2s1 1.2 1.6 1.5c.7.4 1.5.5 2.4.5 1 0 1.8-.2 2.4-.6a4 4 0 0 0 1.6-1.8c.3-.8.5-1.8.5-3V45.5zm30.9 9.1a4.4 4.4 0 0 0-2-3.3 7.5 7.5 0 0 0-4.3-1.1c-1.3 0-2.4.2-3.3.5-.9.4-1.6 1-2 1.6a3.5 3.5 0 0 0-.3 4c.3.5.7.9 1.3 1.2l1.8 1 2 .5 3.2.8c1.3.3 2.5.7 3.7 1.2a13 13 0 0 1 3.2 1.8 8.1 8.1 0 0 1 3 6.5c0 2-.5 3.7-1.5 5.1a10 10 0 0 1-4.4 3.5c-1.8.8-4.1 1.2-6.8 1.2-2.6 0-4.9-.4-6.8-1.2-2-.8-3.4-2-4.5-3.5a10 10 0 0 1-1.7-5.6h6a5 5 0 0 0 3.5 4.6c1 .4 2.2.6 3.4.6 1.3 0 2.5-.2 3.5-.6 1-.4 1.8-1 2.4-1.7a4 4 0 0 0 .8-2.4c0-.9-.2-1.6-.7-2.2a11 11 0 0 0-2.1-1.4l-3.2-1-3.8-1c-2.8-.7-5-1.7-6.6-3.2a7.2 7.2 0 0 1-2.4-5.7 8 8 0 0 1 1.7-5 10 10 0 0 1 4.3-3.5c2-.8 4-1.2 6.4-1.2 2.3 0 4.4.4 6.2 1.2 1.8.8 3.2 2 4.3 3.4 1 1.4 1.5 3 1.5 5h-5.8z"/></svg>
</file>

<file path="client/public/robots.txt">
Allow all web crawlers to index the site
User-agent: *
Allow: /

Sitemap location
Sitemap: https://yourdomain.com/sitemap.xml

Disallow admin and API routes
Disallow: /admin/
Disallow: /api/

Allow media files
Allow: /public/

Crawl-delay: 10 # Uncomment and adjust if needed to limit crawl rate
</file>

<file path="client/public/vercel.svg">
<svg fill="none" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 1155 1000"><path d="m577.3 0 577.4 1000H0z" fill="#fff"/></svg>
</file>

<file path="client/public/window.svg">
<svg fill="none" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 16 16"><path fill-rule="evenodd" clip-rule="evenodd" d="M1.5 2.5h13v10a1 1 0 0 1-1 1h-11a1 1 0 0 1-1-1zM0 1h16v11.5a2.5 2.5 0 0 1-2.5 2.5h-11A2.5 2.5 0 0 1 0 12.5zm3.75 4.5a.75.75 0 1 0 0-1.5.75.75 0 0 0 0 1.5M7 4.75a.75.75 0 1 1-1.5 0 .75.75 0 0 1 1.5 0m1.75.75a.75.75 0 1 0 0-1.5.75.75 0 0 0 0 1.5" fill="#666"/></svg>
</file>

<file path="client/README.md">
This is a [Next.js](https://nextjs.org) project bootstrapped with [`create-next-app`](https://nextjs.org/docs/app/api-reference/cli/create-next-app).

Getting Started

First, run the development server:

```bash
npm run dev
# or
yarn dev
# or
pnpm dev
# or
bun dev
```

Open http://localhost:3000 with your browser to see the result.

You can start editing the page by modifying `app/page.tsx`. The page auto-updates as you edit the file.

This project uses [`next/font`](https://nextjs.org/docs/app/building-your-application/optimizing/fonts) to automatically optimize and load [Geist](https://vercel.com/font), a new font family for Vercel.

Learn More

To learn more about Next.js, take a look at the following resources:

- [Next.js Documentation](https://nextjs.org/docs) - learn about Next.js features and API.
- [Learn Next.js](https://nextjs.org/learn) - an interactive Next.js tutorial.

You can check out [the Next.js GitHub repository](https://github.com/vercel/next.js) - your feedback and contributions are welcome!

Deploy on Vercel

The easiest way to deploy your Next.js app is to use the [Vercel Platform](https://vercel.com/new?utm_medium=default-template&filter=next.js&utm_source=create-next-app&utm_campaign=create-next-app-readme) from the creators of Next.js.

Check out our [Next.js deployment documentation](https://nextjs.org/docs/app/building-your-application/deploying) for more details.
</file>

<file path="client/src/app/about/components/CompanyDescription.tsx">
```
import React from 'react';
import Icon from '@/components/ui/Icon';
import { IoBusinessSharp } from 'react-icons/io5';
import { VscSymbolStructure } from 'react-icons/vsc';
import { RxValue } from 'react-icons/rx';
import Section from '@/components/ui/layout/Section';
const CompanyDescription: React.FC = () => {
      return (
            <Section
                  heading='PA Luxe Creation PTY Ltd was established in 2024 by Precious Nyathi. Precious identified a need
                              in Evander for modern, affordable services that combined convenience and quality. This
                              led to the concept of a dual-service establishment offering both delicious,
                              budget-friendly meals and premium car care.'
                  tittle='Company Description'
```

```
                  Icon={IoBusinessSharp}>
                  <article>
                        <Icon
                              icon={VscSymbolStructure}
                              variant='inlineCircular'
                              heading='Legal Structure'
                        />
                        <p>
                              PA Luxe Creation PTY Ltd is registeyellow as a
Private Company (PTY) in 2023. This structure
                              was chosen to provide limited liability protection
to the owners, establish the business
                              as a separate legal entity, and offer potential
tax advantages. Registration as a PTY also
                              enhances the company&amp;apos;s cyellowibility and
facilitates future fundraising efforts.
                        </p>
                  </article>

                  <article>
                        <Icon
                              icon={RxValue}
                              heading='Unique Value Proposition'
                        />
                        <p className='text-white'>
                              &amp;ldquo;PA Luxe Creation delivers unbeatable
pricing, premium service quality, and
                              cutting-edge convenience through culinary
excellence and industrial engineering.&amp;rdquo;
                        </p>
                  </article>
            </Section>
      );
};

export default CompanyDescription;
</file>

<file path="client/src/app/about/components/CoreValues.tsx">
import { IoBulbOutline, IoShieldCheckmarkOutline, IoPeopleOutline } from 'react-
icons/io5';
import Icon from '@/components/ui/Icon';
import Section from '@/components/ui/layout/Section';

const CoreValues: React.FC = () => {
      return (
            <Section tittle='Our Core Values'>
                  <div className='mt-6 grid grid-cols-1 md:grid-cols-2 gap-2
text-center'>
                        <article>
                              <Icon
                                    icon={IoBulbOutline}
                                    heading='Innovation'
                              />
                              <p>We constantly seek new ways to enhance our
services and customer experience.</p>
                        </article>
                        <article>
                              <Icon
                                    icon={IoShieldCheckmarkOutline}
                                    heading='Quality'
                              />
                              <p>We are committed to providing high-quality food
and exceptional Photo boot services.</p>
```

```
                        </article>
                        <article>
                              <Icon
                                    icon={IoPeopleOutline}
                                    heading='Community Trust'
                              />
                              <p>
                                    We aim to build strong relationships with
our customers and contribute positively to the
                                    Evander community.
                              </p>
                        </article>
                  </div>
            </Section>
      );
};

export default CoreValues;
</file>

<file path="client/src/app/about/components/KeyHighlights.tsx">
// components/KeyHighlights.tsx
import React from 'react';
import { IoBusiness, IoPricetag, IoCalendar } from 'react-icons/io5';
import Icon from '@/components/ui/Icon';
import Section from '@/components/ui/layout/Section';

const KeyHighlights = () => {
      return (
            <Section tittle='Key Highlights of PA Luxe Creation'>
                  <div className='grid grid-cols-1 md:grid-cols-2  gap-2 '>
                        {/* Highlight 1: Unique Combination of Services */}
                        <article>
                              <Icon icon={IoBusiness} />
                              <h3>Unique Service Combination</h3>
                              <p className=' text-center'>
                                    PA Luxe Creation stands out by offering a
convenient combination of a Photo boot and an
                                    eatery, catering to multiple needs in one
stop.
                              </p>
                        </article>

                        {/* Highlight 2: Competitive Photo boot Pricing */}
                        <article>
                              <Icon icon={IoPricetag} />
                              <h3>Competitive Photo boot Pricing</h3>
                              <p className=' text-center'>
                                    We offer attractive and competitive pricing
for our Photo boot services, providing
                                    excellent value for our customers.
                              </p>
                        </article>

                        {/* Highlight 3: Advanced Booking and Modern POS */}
                        <article>
                              <Icon icon={IoCalendar} />
                              <h3>Advanced Systems</h3>
                              <p className=' text-center'>
                                    Utilizing an advanced booking system and
modern POS integration for a seamless and
                                    efficient customer experience.
                              </p>
                        </article>
```

```
                </div>
            </Section>
      );
};

export default KeyHighlights;
</file>

<file path="client/src/app/about/components/KeyLeadership.tsx">
import { IoPeopleOutline, IoCheckmarkCircleOutline } from 'react-icons/io5';
import Icon from '@/components/ui/Icon';
import Section from '@/components/ui/layout/Section';

const KeyLeadership: React.FC = () => {
      return (
            <Section
                  tittle='Key Leadership'
                  Icon={IoPeopleOutline}
                  heading='PA Luxe Creation is driven by a dedicated leadership
team with a diverse range of expertise.
                              A key asset to our team is A.E.N Nyathi, an
accomplished Industrial Engineer and
                              Full-Stack Designer.'>
                  <article className='leading-relaxed p-4 mt-4 text-base
md:text-lg'>
                        <p>A.E.N Nyathi brings a wealth of knowledge and skills
to PA Luxe Creation, including:</p>
                        <div className='grid grid-cols-1 md:grid-cols-2 gap-4
mt-4'>
                              {[
                                    'Graphic Design (Photoshop, Illustrator)',
                                    'Web Development (full stack)',
                                    'Accounting & Electrical Systems',
                                    'Operational Workflow Optimization (Fusion
360)',
                                    'AI Agents (n8n)',
                              ].map((skill) => (
                                    <div
                                          key={skill}
                                          className='flex items-center gap-2'>
                                          <Icon
                                                icon={IoCheckmarkCircleOutline}
                                                variant='inline'
                                                className='text-xl text-
yellow-500 flex-shrink-0'
                                          />
                                          <span>{skill}</span>
                                    </div>
                              ))}
                        </div>
                        <p className='mt-4'>
                              This unique blend of technical and creative
expertise is instrumental in driving Central
                              Eatery&amp;apos;s success, from implementing
innovative technology solutions to optimizing our
                              operational workflows. A.E.N Nyathi&amp;apos;s
contributions ensure that PA Luxe Creation
                              remains at the forefront of convenience and
quality in Evander.
                        </p>
                  </article>
            </Section>
      );
};
```

```
export default KeyLeadership;
</file>

<file path="client/src/app/about/components/MilestonesAndVision.tsx">
import Icon from '@/components/ui/Icon';
import Section from '@/components/ui/layout/Section';
import { IoFlag, IoRocketOutline } from 'react-icons/io5';

const MilestonesAndVision: React.FC = () => {
      return (
            <Section
                  heading='Our journey began with a vision to transform
convenience in Evander. Here are some key
                        moments and our aspirations for the future.'
                  tittle='Milestones & Future Vision'>
                  <article className='mt-6'>
                        <Icon
                              icon={IoFlag}
                              variant='inlineCircular'
                              className='text-3xl text-yellow-500'
                              heading='Key Milestones'
                        />

                        <ul className='list-disc list-inside '>
                              <li>
                                    <strong>2024:</strong> PA Luxe Creation PTY
Ltd was established, bringing together
                                    affordable, high-quality meals.
                              </li>
                              <li>
                                    <strong>2025:</strong> PA Luxe Creation PTY
Ltd has established a new building in Evander
                                    which included luxury Photo boot.
                              </li>

                              {/* Add future milestones here */}
                        </ul>
                  </article>

                  <article className='mt-6'>
                        <Icon
                              icon={IoRocketOutline}
                              variant='inlineCircular'
                              heading='Our Future Vision'
                        />
                        <p className='text-center'>
                              Our vision is to become Evander&amp;apos;s premier
destination for these combined services,
                              driven by innovation, quality, and community
trust. We aim to continuously innovate,
                              expand our offerings, and deepen our connection
with the community we serve.
                        </p>
                  </article>
            </Section>
      );
};

export default MilestonesAndVision;
</file>

<file path="client/src/app/about/components/OurCommitment.tsx">
import Section from '@/components/ui/layout/Section';
```

```
import Image from 'next/image';
import { IoShieldCheckmark } from 'react-icons/io5';

const OurCommitment: React.FC = () => {
      return (
            <Section
                  Icon={IoShieldCheckmark}
                  heading='At PA Luxe Creation PTY Ltd, we are committed to
providing a premium experience in both our
                              food service and car care offerings. We strive to
be your convenient and affordable
                              destination, delivering quality meals and
luxurious Photo boot. Our goal is to build
                              community trust through innovation and exceptional
service in Evander.'
                  tittle='Our Commitment'>
                  <Image
                        alt='logo'
                        height={300}
                        width={300}
                        src='/PA_Logo.png'
                  />
            </Section>
      );
};

export default OurCommitment;
</file>

<file path="client/src/app/about/components/OurJourney.tsx">
import Section from '@/components/ui/layout/Section';
import Image from 'next/image';

const OurJourney: React.FC = () => {
      return (
            <Section tittle='Our Journey to PA Luxe Creation'>
                  <Image
                        src='/PA_Logo.png'
                        alt='logo'
                        width={150}
                        height={150}
                        className='float-right'
                  />
                  <p className='text-start'>
                        PA Luxe Creation PTY Ltd was established in 2024 by
Precious Nyathi. Nyathi identified a need in
                        Evander for modern, affordable services that combined
convenience and quality. This vision
                        led to the concept of a dual-service establishment
offering both delicious, budget-friendly
                        meals and premium car care. Our journey is fueled by a
commitment to providing a unique and
                        valuable service to our community.
                  </p>
            </Section>
      );
};

export default OurJourney;
</file>

<file path="client/src/app/about/components/OurVision.tsx">
import Icon from '@/components/ui/Icon';
import Section from '@/components/ui/layout/Section';
```

```
import { IoEye } from 'react-icons/io5';

const OurVision: React.FC = () => {
      return (
            <Section
                  Icon={IoEye}
                  tittle='Our Vision'>
                  <article>
                        <p className='text-center '>
                              <strong>Our vision</strong> is to become
Evander&amp;apos;s premier destination for
                              affordable, high-quality meals and competitively
priced luxury Photo boot, driven by
                              innovation, quality, and community trust.
                        </p>
                  </article>
            </Section>
      );
};

export default OurVision;
</file>

<file path="client/src/app/about/components/StrengthsAndOpportunities.tsx">
import Icon from '@/components/ui/Icon';
import Section from '@/components/ui/layout/Section';
import React from 'react';
import { GiMuscleUp, GiOppositeHearts } from 'react-icons/gi';

const StrengthsAndOpportunities: React.FC = () => {
      const strengths = [
            '20â “30% cheaper Photo boot',
            'multidisciplinary leadership',
            'dual-service convenience.',
      ];

      const opportunities = ['Taxi fleet bulk discounts,', 'loyalty program
integration.'];

      return (
            <Section tittle='Strengths and Opportunities'>
                  <div className='mt-6 grid grid-cols-1 md:grid-cols-2 gap-2
'>
                        <article>
                              <Icon
                                    icon={GiMuscleUp}
                                    heading='Strengths'
                              />
                              <ul className='list-disc list-inside '>
                                    {strengths.map((strength, index) => (
                                          <li key={index}>{strength}</li>
                                    ))}
                              </ul>
                        </article>

                        <article>
                              <Icon
                                    icon={GiOppositeHearts}
                                    heading='Opportunities'
                              />
                              <ul className='list-disc list-inside '>
                                    {opportunities.map((opportunity, index) => (
                                          <li key={index}>{opportunity}</li>
                                    ))}
```

```
                        </ul>
                    </article>
                </div>
            </Section>
        );
};

export default StrengthsAndOpportunities;
</file>

<file path="client/src/app/about/components/WeaknessesAndThreats.tsx">
import Section from '@/components/ui/layout/Section';
import React from 'react';

const WeaknessesAndThreats: React.FC = () => {
      const weaknesses = ['Reliant on local foot traffic.', 'New brand
awareness.', 'Limited funding.'];

      const threats = ['Competitor price matching', 'economic downturns',
'supply chain disruptions'];

      return (
            <Section tittle='Weaknesses and Threats'>
                  <div className='mt-6 grid grid-cols-1 md:grid-cols-2 gap-2
'>
                        <article>
                              <h3>Weaknesses</h3>
                              <ul className='list-disc list-inside '>
                                    {weaknesses.map((weakness, index) => (
                                          <li key={index}>{weakness}</li>
                                    ))}
                              </ul>
                        </article>
                        <article>
                              <h3>Threats</h3>
                              <ul className='list-disc list-inside '>
                                    {threats.map((threat, index) => (
                                          <li key={index}>{threat}</li>
                                    ))}
                              </ul>
                        </article>
                  </div>
            </Section>
      );
};

export default WeaknessesAndThreats;
</file>

<file path="client/src/app/about/components/WhatMakesUsSpecial.tsx">
import { IoSparkles, IoCalendar, IoPeople, IoFastFood } from 'react-icons/io5';
import Icon from '@/components/ui/Icon';
import Section from '@/components/ui/layout/Section';

const WhatMakesUsSpecial: React.FC = () => {
      return (
            <Section tittle='What Makes Us Special?'>
                  <div className='grid grid-cols-1 md:grid-cols-2  gap-2 gap-2
'>
                        <article>
                              <Icon
                                    icon={IoSparkles}
                                    heading='Modern Facilities'
                              />
```

```
                              <p className='text-center'>
                                    Unlike competitors, PA Luxe Creation boasts
up-to-date facilities for both our eatery and
                                    Photo boot, ensuring a higher standard of
service and a more comfortable experience for
                                    our customers.
                              </p>
                        </article>
                        <article>
                              <Icon
                                    icon={IoCalendar}
                                    heading='Digital Booking System'
                              />

                              <p className='text-center'>
                                    We offer a convenient digital booking
system, allowing you to easily schedule Photo boot
                                    and potentially food orders, saving you time
and providing a seamless experience that
                                    our competitors don&apos;t.
                              </p>
                        </article>
                        <article>
                              <Icon
                                    icon={IoPeople}
                                    heading='Multidisciplinary Expertise'
                              />
                              <p className='text-center'>
                                    Our leadership team possesses diverse
expertise, from industrial engineering to design
                                    and business management, enabling us to
optimize operations, enhance customer
                                    experience, and innovate in ways our
competitors cannot.
                              </p>
                        </article>
                        <article>
                              <Icon
                                    icon={IoFastFood}
                                    heading='Dual-Service Convenience'
                              />
                              <p className='text-center'>
                                    We offer the unique convenience of a high-
quality eatery and a luxury Photo boot in one
                                    location, saving you time and effort
compayellow to visiting separate establishments.
                              </p>
                        </article>
                  </div>
            </Section>
      );
};

export default WhatMakesUsSpecial;
</file>

<file path="client/src/app/blog/components/BlogPostList.tsx">
import React from 'react';

const posts = [
      {
            title: 'The Secret to Our Famous Kota',
            date: 'July 24, 2025',
            excerpt:
```

```
                  'Discover the fresh ingyellowients and secret spices that make
our Kotas the talk of the town...',
      },
      {
            title: '5 Tips for a Sparkling Clean Car',
            date: 'July 20, 2025',
            excerpt: 'Learn our top tips for maintaining a pristine car between
professional washes...',
      },
      {
            title: 'Community Day at PA Luxe Creation',
            date: 'July 15, 2025',
            excerpt:
                  'A look back at our recent community event, filled with good
food and great company...',
      },
];

const BlogPostList: React.FC = () => {
      return (
            <div className='grid grid-cols-1 md:grid-cols-2  gap-6'>
                  {posts.map((post, index) => (
                        <div
                              key={index}
                              className='bg-black/50 p-6 rounded-md shadow-lg
hover:shadow-yellow-500/50 transition-shadow duration-300'>
                              <h3 className='text-xl font-bold
mb-2'>{post.title}</h3>
                              <p className='text-sm text-white mb-2'>{post.date}
</p>
                              <p className='text-white'>{post.excerpt}</p>
                        </div>
                  ))}
            </div>
      );
};

export default BlogPostList;
</file>

<file path="client/src/app/blog/loading.tsx">
import Loading from '@/components/ui/Loading';

export default function LoadingPage() {
      return <Loading message='Loading Blog Posts...' />;
}
</file>

<file path="client/src/app/contact/components/OperatingHours.tsx">
import Section from '@/components/ui/layout/Section';

const OperatingHours: React.FC = () => {
      return (
            <Section tittle='Operating Hours:'>
                  <div className='mt-2 space-y-4'>
                        <article>
                              <h3 className='text-lg font-semibold'>Eatery</h3>
                              <ul>
                                    <li>
                                          <strong>Monday - Friday:</strong> 8:00
AM - 8:00 PM
                                    </li>
                                    <li>
                                          <strong>Saturday:</strong> 8:00 AM -
```

```
9:00 PM
                                    </li>
                                    <li>
                                          <strong>Sunday:</strong> 8:00 AM -
8:00 PM
                                    </li>
                              </ul>
                        </article>
                        <article>
                              <h3 className='text-lg font-semibold'>Photo
boot</h3>
                              <ul>
                                    <li>
                                          <strong>Monday - Friday:</strong> 8:00
AM - 8:00 PM
                                    </li>
                                    <li>
                                          <strong>Saturday:</strong> 8:00 AM -
6:00 PM
                                    </li>
                                    <li>
                                          <strong>Sunday:</strong> Closed
                                    </li>
                              </ul>
                        </article>
                  </div>
            </Section>
      );
};

export default OperatingHours;
</file>

<file path="client/src/app/contact/loading.tsx">
import Loading from '@/components/ui/Loading';

export default function LoadingPage() {
      return <Loading message='Loading Contact Page...' />;
}
</file>

<file path="client/src/app/gallery/components/ImageModal.tsx">
import SocialButtons from '@/app/menu/[Products]/components/SocialButtons';

import { NextPage } from 'next';
import Image from 'next/image';

interface Props {
      handleBackdropClick: (e: React.MouseEvent<HTMLDivElement>) => void;
      product: ProductType;
      setModalOpen: React.Dispatch<React.SetStateAction<boolean>>;
}

const ImageModal: NextPage<Props> = ({ handleBackdropClick, setModalOpen,
product }) => {
      return (
            <main
                  className='fixed inset-0 z-50 flex items-center justify-center
bg-black  animate-fadein p-4'
                  onClick={handleBackdropClick}
                  aria-modal='true'
                  role='dialog'>
                  <div className='relative max-w-4xl  w-full '>
                        <button
```

```
                              onClick={() => setModalOpen(false)}
                              className='absolute top-3 right-3 bg-yellow-600
hover:bg-yellow-700 text-white rounded-full p-2 shadow-lg focus:outline-none
focus-visible:ring-2 focus-visible:ring-white z-10'
                              aria-label='Close full image'
                              autoFocus>
                              <svg
                                    width='24'
                                    height='24'
                                    fill='none'
                                    viewBox='0 0 24 24'>
                                    <path
                                          stroke='currentColor'
                                          strokeWidth='2'
                                          strokeLinecap='round'
                                          strokeLinejoin='round'
                                          d='M6 6l12 12M6 18L18 6'
                                    />
                              </svg>
                        </button>
                        <article className='w-full relative m-0 p-0  flex flex-
col items-center justify-center'>
                              {product?.Image && (
                                    <div className='relative p-4 w-full max-h-
[75vh] '>
                                          <Image
                                                src={product.Image}
                                                alt={`${product.Name} gallery
image`}
                                                layout='responsive'
                                                width={500}
                                                height={500}
                                                className=' rounded-md'
                                                priority
                                          />
                                    </div>
                              )}
                              <div className='text-white text-center text-lg
font-bold'>{product.Name}</div>
                              <SocialButtons product={product} />
                        </article>
                  </div>
            </main>
      );
};

export default ImageModal;
</file>

<file path="client/src/app/gallery/loading.tsx">
import Loading from '@/components/ui/Loading';

export default function LoadingPage() {
      return <Loading message='Loading Gallery...' />;
}
</file>

<file path="client/src/app/home-components/AboutUsSnippet.tsx">
import Icon from '@/components/ui/Icon';
import Section from '@/components/ui/layout/Section';
import AppLink from '@/components/ui/Link';
import { IoInformationCircleOutline } from 'react-icons/io5';

const AboutUsSnippet: React.FC = () => {
```

```
      return (
            <Section
                  Icon={IoInformationCircleOutline}
                  tittle='About PA Luxe Creation'
                  heading='PA Luxe Creation aims to revolutionize convenience in
Evander by offering both affordable, high-quality meals and competitively priced
luxury Photo boot under one roof. Founded with a vision to be the premier
destination for these combined services, we are driven by innovation, quality,
and community trust.'>
                  <div className='mt-6'>
                        <AppLink
                              variant='primary'
                              href='/about'>
                              Read More About Us...
                        </AppLink>
                  </div>
            </Section>
      );
};

export default AboutUsSnippet;
</file>

<file path="client/src/app/home-components/ContactSection.tsx">
import MapEmbed from '@/app/contact/components/MapEmbed';
import { IoLocationSharp, IoCall, IoChatbubblesOutline } from 'react-icons/io5';
import Icon from '@/components/ui/Icon';
import ContactInfo from '@/app/contact/components/ContactInfo';
import Section from '@/components/ui/layout/Section';

const ContactSection: React.FC = () => {
      return (
            <>
                  <ContactInfo />
                  <Section
                        Icon={IoChatbubblesOutline}
                        tittle='Visit Us Today!'>
                        <div className='mt-8'>
                              <MapEmbed />
                        </div>
                  </Section>
            </>
      );
};

export default ContactSection;
</file>

<file path="client/src/app/home-components/FinancialSnapshotTeaser.tsx">
import Section from '@/components/ui/layout/Section';
import Link from '@/components/ui/Link';
import { IoTrendingUpOutline } from 'react-icons/io5';

const FinancialSnapshotTeaser: React.FC = () => {
      return (
            <Section
                  Icon={IoTrendingUpOutline}
                  tittle='Building a Sustainable Future'
                  heading="Since our inception, we've seen promising traction
and are on a solid path toward sustainable growth. Our commitment to quality and
convenience continues to drive our success.">
                  <div className='mt-6'>
                        <Link
                              variant='button'
```

```
                              href='/about#financials'>
                              View Our Financial Snapshot...
                        </Link>
                  </div>
            </Section>
      );
};

export default FinancialSnapshotTeaser;
</file>

<file path="client/src/app/home-components/KeyDifferentiators.tsx">
import Icon from '@/components/ui/Icon';
import Section from '@/components/ui/layout/Section';
import React from 'react';
import { GiFoodTruck } from 'react-icons/gi';
import { HiHomeModern } from 'react-icons/hi2';
import { RiMoneyDollarCircleLine } from 'react-icons/ri';

const KeyDifferentiators: React.FC = () => {
      return (
            <Section tittle='What Makes PA Luxe Creation Special?'>
                  <div className='grid grid-cols-1 md:grid-cols-2 gap-2'>
                        {/* Unique Combination */}
                        <article>
                              <Icon
                                    icon={GiFoodTruck}
                                    heading='Food & Car Care in One'
                              />
                              <p className='mt-2'>
                                    Revolutionizing convenience with delicious,
affordable meals and premium Photo boot
                                    services under one roof.
                              </p>
                        </article>

                        {/* Competitive Pricing */}
                        <article>
                              <Icon
                                    icon={RiMoneyDollarCircleLine}
                                    heading='Unbeatable Value'
                              />
                              <p className='mt-2'>
                                    Enjoy competitively priced Photo boot (20-
30% cheaper!) and high-quality,
                                    budget-friendly meals.
                              </p>
                        </article>

                        {/* Modern Technology */}
                        <article>
                              <Icon
                                    icon={HiHomeModern}
                                    heading='Modern Convenience'
                              />
                              <p className='mt-2'>
                                    Benefit from an advanced booking system for
Photo boot and a modern POS system for a
                                    smooth experience.
                              </p>
                        </article>
                  </div>
            </Section>
      );
```

```
};

export default KeyDifferentiators;
</file>

<file path="client/src/app/home-components/KeyHighlights.tsx">
import Section from '@/components/ui/layout/Section';
import { IoSparklesOutline, IoCheckmarkCircle } from 'react-icons/io5';

const KeyHighlights: React.FC = () => {
      const highlights = [
            {
                  id: 1,
                  text: 'Dual-Service Convenience: Enjoy a tasty meal while your
car is being taken care of.',
            },
            { id: 2, text: "Affordability: Quality food and Photo boot services
that won't break the bank." },
            {
                  id: 3,
                  text: "Community-Centric: We're dedicated to serving and
supporting the Evander community.",
            },
      ];

      return (
            <Section
                  Icon={IoSparklesOutline}
                  tittle='The PA Luxe Creation Difference'>
                  <div className=' grid md:grid-cols-2 gap-8'>
                        {highlights.map((highlight) => (
                              <div
                                    key={highlight.id}
                                    className='flex flex-col items-center'>
                                    <IoCheckmarkCircle className='text-4xl text-
yellow-500 mb-3' />
                                    <p>{highlight.text}</p>
                              </div>
                        ))}
                  </div>
            </Section>
      );
};

export default KeyHighlights;
</file>

<file path="client/src/app/home-components/TargetMarketCallout.tsx">
import { BiBullseye } from 'react-icons/bi';
import {
      IoHomeOutline,
      IoBriefcaseOutline,
      IoCarSportOutline,
      IoBusinessOutline,
} from 'react-icons/io5';
import Icon from '@/components/ui/Icon';
import Section from '@/components/ui/layout/Section';

const TargetMarketCallout: React.FC = () => {
      const targetMarkets = [
            {
                  id: 1,
                  title: 'Residents of Evander',
                  text: 'Catering to families, professionals, and students
```

```
seeking convenient, high-quality meals and car care.',
                  icon: IoHomeOutline,
            },
            {
                  id: 2,
                  title: 'People Who Work in the Area',
                  text: 'Providing quick meal solutions and efficient Photo boot
services for busy professionals and government workers.',
                  icon: IoBriefcaseOutline,
            },
            {
                  id: 3,
                  title: 'Travelers Passing Through',
                  text: 'A welcoming stop for travelers to rest and refresh with
reliable meals and clean Photo boot services.',
                  icon: IoCarSportOutline,
            },
            {
                  id: 4,
                  title: 'Local Businesses',
                  text: 'Partnering with local businesses for catering services
and fleet vehicle maintenance programs.',
                  icon: IoBusinessOutline,
            },
      ];

      return (
            <Section
                  Icon={BiBullseye}
                  tittle='Our Target Market'
                  id='target-market'
                  heading='PA Luxe Creation is dedicated to serving the diverse
needs of the Evander community and those passing through.'>
                  <div className='grid grid-cols-1 md:grid-cols-2  gap-2'>
                        {targetMarkets.map((market) => (
                              <article
                                    key={market.id}
                                    className='flex flex-col items-center'>
                                    <Icon icon={market.icon} />
                                    <h3>{market.title}</h3>
                                    <p className='text-yellow-100 text-
center'>{market.text}</p>
                              </article>
                        ))}
                  </div>
            </Section>
      );
};

export default TargetMarketCallout;
</file>

<file path="client/src/app/loading.tsx">
import Loading from '@/components/ui/Loading';

export default function LoadingPage() {
      return <Loading message='Loading Home page...' />;
}
</file>

<file path="client/src/app/menu/[Products]/[product]/comments/loading.tsx">
import Loading from '@/components/ui/Loading';

export default function LoadingPage() {
```

```
      return <Loading message='Loading Product Comments...' />;
}
</file>

<file path="client/src/app/menu/[Products]/[product]/loading.tsx">
import Loading from '@/components/ui/Loading';

export default function LoadingPage() {
      return <Loading message='Loading Product Details...' />;
}
</file>

<file path="client/src/app/menu/[Products]/loading.tsx">
import Loading from '@/components/ui/Loading';

export default function LoadingPage() {
      return <Loading message='Loading Products...' />;
}
</file>

<file path="client/src/app/menu/cart/loading.tsx">
import Loading from '@/components/ui/Loading';

export default function LoadingPage() {
      return <Loading message='Loading Cart...' />;
}
</file>

<file path="client/src/app/menu/components/CategoryCard.tsx">
import React from 'react';
import Image from 'next/image';
import AppLink from '@/components/ui/Link';

import Section from '@/components/ui/layout/Section';

interface CategoryCardProps {
      group: {
            Name: string;
            Products: ProductType[];
            Image?: string;
            Description?: string;
      };
}

const CategoryCard: React.FC<CategoryCardProps> = ({ group }) => (
      <AppLink
            href={`/menu/${group.Name.toLowerCase().replace(/\s+/g, '-')}`}
            aria-label={`View ${group.Name} menu`}
            className='block p-0 m-0 group rounded-md focus:outline-none
focus:ring-2 focus:ring-offset-2 focus:ring-offset-black/50 focus:ring-
yellow-500'>
            <Section>
                  {group.Image && (
                        <div className='relative w-full aspect-[4/3] mb-4 bg-
black/20 rounded-md overflow-hidden'>
                              <Image
                                    src={group.Image}
                                    alt={group.Name}
                                    fill
                                    className='object-contain rounded-md
transition-transform duration-300 group-hover:scale-110'
                              />
                        </div>
                  )}
```

```
                  <div className='flex-grow flex flex-col'>
                        <h3 className='text-xl font-bold text-white
mb-2'>{group.Name}</h3>
                        {group.Description && <p>{group.Description}</p>}
                  </div>
                  <span className='bg-yellow-700 text-xs font-bold px-3 py-1
rounded-full text-white self-center mt-auto'>
                        {group.Products.length} items
                  </span>
            </Section>
      </AppLink>
);

export default CategoryCard;
</file>

<file path="client/src/app/menu/components/NoResults.tsx">
import React from 'react';
import { FaRegSadTear } from 'react-icons/fa';

const NoResults: React.FC = () => (
      <div className='col-span-full flex flex-col items-center justify-center
text-center text-white p-8'>
            <div className='bg-black/50 border border-white/50 rounded-md p-8
max-w-md w-full'>
                  <FaRegSadTear className='text-6xl mb-4 text-yellow-500 mx-
auto' />
                  <h3 className='text-2xl font-bold'>No Results Found</h3>
                  <p className='text-white mt-2'>
                        We couldn&apos;t find any matches for your search.
Please try different keywords or browse
                        our menu categories.
                  </p>
            </div>
      </div>
);

export default NoResults;
</file>

<file path="client/src/app/menu/components/ProductDetails.tsx">
import Image from 'next/image';
import React from 'react';

function ProductDetails({ item }: { item: CartItem }) {
      return (
            <div className='flex items-center gap-4 w-full text-white
bg-black/20  rounded-md p-4'>
                  {item.Image && (
                        <div className=' w-20 h-20 sm:w-24 sm:h-24  flex-
shrink-0'>
                              <Image
                                    src={item.Image}
                                    alt={item.Name}
                                    height={100}
                                    width={100}
                                    className='w-full rounded-md  '
                              />
                        </div>
                  )}
                  <div className='flex-grow flex flex-col justify-between h-
full'>
                        <h4>{item.Name}</h4>
                        <div className='space-y-1 text-sm'>
```

```
                              <div className='flex justify-between'>
                                    <span
className='text-white'>Quantity:</span>
                                    <span className='font-
semibold'>{item.quantity}</span>
                              </div>
                              <div className='flex justify-between'>
                                    <span className='text-white'>Price:</span>
                                    <span className='font-
semibold'>R{item.Price.toFixed(2)}</span>
                              </div>
                              {item.quantity > 1 && (
                                    <div className='flex justify-between font-
bold border-t border-white/20 pt-1 mt-1'>
                                          <span
className='text-white'>Total:</span>
                                          <span>R{(item.Price *
item.quantity).toFixed(2)}</span>
                                    </div>
                              )}
                        </div>
                  </div>
            </div>
      );
}

export default ProductDetails;
</file>

<file path="client/src/app/menu/components/PromotionsBanner.tsx">
'use client';
import React, { useState, useEffect } from 'react';
import { FaGift, FaCar, FaHamburger } from 'react-icons/fa';

const promotions = [
      {
            icon: <FaGift className='text-4xl text-white drop-shadow-lg' />,
            title: 'Weekend Special!',
            message: "Free pork plate with every Photo boot (Sat/Sun). Don't
miss out!",
      },
      {
            icon: <FaCar className='text-4xl text-white drop-shadow-lg' />,
            title: 'Taxi Fleet Discount',
            message: 'Bulk taxi washes: Buy 10, get 1 free! Perfect for
operators.',
      },
      {
            icon: <FaHamburger className='text-4xl text-white drop-shadow-lg'
/>,
            title: 'Meal Loyalty',
            message: 'Buy 10 meals, get a free Photo boot. Ask for your digital
loyalty card!',
      },
];

const PromotionsBanner: React.FC = () => {
      const [index, setIndex] = useState(0);
      const [fade, setFade] = useState(true);

      useEffect(() => {
            const timer = setTimeout(() => {
                  setFade(false);
                  setTimeout(() => {
```

```
                        setIndex((prev) => (prev + 1) % promotions.length);
                        setFade(true);
                  }, 500); // Corresponds to transition duration
            }, 6000);
            return () => clearTimeout(timer);
      }, [index]);

      const promo = promotions[index];

      const handleNavClick = (i: number) => {
            if (i === index) return;
            setFade(false);
            setTimeout(() => {
                  setIndex(i);
                  setFade(true);
            }, 500);
      };

      return (
            <div className='bg-black/50 blur-[0.1px] backdrop-blur-md p-4 sticky
bottom-0 left-0 w-full z-10 border-t border-white/20'>
                  <div className='w-full max-w-3xl mx-auto flex flex-col items-
center'>
                        <div
                              className={`transition-opacity duration-500 ${fade
? 'opacity-100' : 'opacity-0'}`}
                              key={index}>
                              <div className='flex flex-col sm:flex-row items-
center text-center sm:text-left gap-4'>
                                    <div className='flex-shrink-0'>{promo.icon}
</div>
                                    <div>
                                          <h3
                                                className='text-xl font-bold
text-white'
                                                style={{ filter: 'drop-shadow(0 0
3px #f0b100)' }}>
                                                {promo.title}
                                          </h3>
                                          <p className='text-white mt-1 text-sm
sm:text-base'>{promo.message}</p>
                                    </div>
                              </div>
                        </div>
                        <div className='flex justify-center w-full mt-3'>
                              <div className='flex gap-2.5'>
                                    {promotions.map((_, i) => (
                                          <button
                                                key={i}
                                                className={`w-2.5 h-2.5 rounded-
full border border-white/50 transition-all duration-300 focus:outline-none
focus:ring-2 focus:ring-offset-2 focus:ring-offset-black focus:ring-yellow-500 $
{
                                                      i === index ? 'bg-white
scale-110' : 'bg-transparent hover:bg-white/30'
                                                }`}
                                                aria-label={`Show promotion ${i +
1}`}
                                                onClick={() => handleNavClick(i)}
                                          />
                                    ))}
                              </div>
                        </div>
                  </div>
```

```
            </div>
        );
};

export default PromotionsBanner;
</file>

<file path="client/src/app/menu/loading.tsx">
import Loading from '@/components/ui/Loading';

export default function LoadingPage() {
      return <Loading message='Loading Menu...' />;
}
</file>

<file path="client/src/app/photo/placeholder-images.json">
{
  "placeholderImages": [
    {
      "id": "hero",
      "description": "A glamorous party scene with people celebrating.",
      "imageUrl": "https://images.unsplash.com/photo-1425421598808-4a22ce59cc97?
crop=entropy&cs=tinysrgb&fit=max&fm=jpg&ixid=M3w3NDE5ODJ8MHwxfHNlYXJjaHw2fHxwYXJ
0eSUyMGV2ZW50fGVufDB8fHx8MTc2MTE0Mjk4MHww&ixlib=rb-4.1.0&q=80&w=1080",
      "imageHint": "party event"
    },
    {
      "id": "service-360-booth",
      "description": "A 360 photo booth in action at an event.",
      "imageUrl": "https://images.unsplash.com/photo-1727892349109-54d06e3478ee?
crop=entropy&cs=tinysrgb&fit=max&fm=jpg&ixid=M3w3NDE5ODJ8MHwxfHNlYXJjaHw3fHxwaG9
0byUyMGJvb3RofGVufDB8fHx8MTc2MTEzOTcxOHww&ixlib=rb-4.1.0&q=80&w=1080",
      "imageHint": "photo booth"
    },
    {
      "id": "service-red-carpet",
      "description": "A red carpet entrance with stanchions.",
      "imageUrl": "https://images.unsplash.com/photo-1614115866447-c9a299154650?
crop=entropy&cs=tinysrgb&fit=max&fm=jpg&ixid=M3w3NDE5ODJ8MHwxfHNlYXJjaHwzfHxyZWQ
lMjBjYXJwZXR8ZW58MHx8fHwxNzYxMTQyOTgwfDA&ixlib=rb-4.1.0&q=80&w=1080",
      "imageHint": "red carpet"
    },
    {
      "id": "event-graduations",
      "description": "Graduates celebrating at a party.",
      "imageUrl": "https://images.unsplash.com/photo-1533120921505-7f40f5237ee1?
crop=entropy&cs=tinysrgb&fit=max&fm=jpg&ixid=M3w3NDE5ODJ8MHwxfHNlYXJjaHw3fHxncmF
kdWF0aW9uJTIwcGFydHl8ZW58MHx8fHwxNzYxMTQyOTgwfDA&ixlib=rb-4.1.0&q=80&w=1080",
      "imageHint": "graduation party"
    },
    {
      "id": "event-weddings",
      "description": "A wedding reception with guests dancing.",
      "imageUrl": "https://images.unsplash.com/photo-1723832348105-2e69f948135a?
crop=entropy&cs=tinysrgb&fit=max&fm=jpg&ixid=M3w3NDE5ODJ8MHwxfHNlYXJjaHw1fHx3ZWR
kaW5nJTIwcmVjZXB0aW9ufGVufDB8fHx8MTc2MTEyMTEyOXww&ixlib=rb-4.1.0&q=80&w=1080",
      "imageHint": "wedding reception"
    },
    {
      "id": "event-baby-showers",
      "description": "Decorations at a baby shower.",
      "imageUrl": "https://images.unsplash.com/photo-1594150878496-a921e5af8907?
crop=entropy&cs=tinysrgb&fit=max&fm=jpg&ixid=M3w3NDE5ODJ8MHwxfHNlYXJjaHw2fHxiYWJ
5JTIwc2hvd2VyfGVufDB8fHx8MTc2MTA1NjI3OHww&ixlib=rb-4.1.0&q=80&w=1080",
```

```
      "imageHint": "baby shower"
    },
    {
      "id": "event-school-events",
      "description": "Students at a school dance.",
      "imageUrl": "https://images.unsplash.com/photo-1504609813442-a8924e83f76e?
crop=entropy&cs=tinysrgb&fit=max&fm=jpg&ixid=M3w3NDE5ODJ8MHwxfHNlYXJjaHwyfHxzY2h
vb2wlMjBkYW5jZXxlbnwwfHx8fDE3NjExNDI5ODN8MA&ixlib=rb-4.1.0&q=80&w=1080",
      "imageHint": "school dance"
    },
    {
      "id": "event-birthdays",
      "description": "A lively birthday party.",
      "imageUrl": "https://images.unsplash.com/photo-1556125574-d7f27ec36a06?
crop=entropy&cs=tinysrgb&fit=max&fm=jpg&ixid=M3w3NDE5ODJ8MHwxfHNlYXJjaHw5fHxiaXJ
0aGRheSUyMHBhcnR5fGVufDB8fHx8MTc2MTA0MTM0N3ww&ixlib=rb-4.1.0&q=80&w=1080",
      "imageHint": "birthday party"
    }
  ]
}
</file>

<file path="client/src/app/photo/placeholderImages.ts">
import data from './placeholder-images.json';

export type ImagePlaceholder = {
  id: string;
  description: string;
  imageUrl: string;
  imageHint: string;
};

export const PlaceHolderImages: ImagePlaceholder[] = data.placeholderImages;
</file>

<file path="client/src/app/privacy/loading.tsx">
const Loading: React.FC = () => {
  return (
    <main>
      Loading Privacy Policy...
    </main>
  );
};

export default Loading;
</file>

<file path="client/src/app/profile/loading.tsx">
import Loading from '@/components/ui/Loading';

export default function LoadingPage() {
      return <Loading message='Loading Profile...' />;
}
</file>

<file path="client/src/app/terms/loading.tsx">
const Loading: React.FC = () => {
      return <div>Loading Terms and Conditions...</div>;
};

export default Loading;
</file>

<file path="client/src/components/Navbar/MobileNavbar.tsx">
```

```
'use client';
import React, { useState, useRef, useEffect, Dispatch, SetStateAction } from
'react';
import Button from '@/components/ui/Button';

const MobileNavbar: React.FC<{
      setMenubar: (path: 'mobile' | 'profile') => void;
      mobileOpen: boolean;
}> = ({ mobileOpen, setMenubar }) => {
      return (
            <nav className='md:hidden w-full m-0 z-50 p-0'>
                  <div className='flex flex-row items-center justify-between w-
full '>
                        <Button
                              variant='icon'
                              onClick={() => {
                                    setMenubar('mobile');
                              }}
                              title={mobileOpen ? 'Close menu' : 'Open menu'}
                              aria-label={mobileOpen ? 'Close menu' : 'Open
menu'}>
                              <svg
                                    className='h-6 w-6'
                                    fill='none'
                                    stroke='currentColor'
                                    viewBox='0 0 24 24'
                                    xmlns='http://www.w3.org/2000/svg'>
                                    {mobileOpen ? (
                                          <path
                                                strokeLinecap='round'
                                                strokeLinejoin='round'
                                                strokeWidth='2'
                                                d='M6 18L18 6M6 6l12 12'
                                          />
                                    ) : (
                                          <path
                                                strokeLinecap='round'
                                                strokeLinejoin='round'
                                                strokeWidth='2'
                                                d='M4 6h16M4 12h16m-7 6h7'
                                          />
                                    )}
                              </svg>
                        </Button>
                  </div>
            </nav>
      );
};

export default MobileNavbar;
</file>

<file path="client/src/components/Navbar/useMenubarToggle.ts">
import { useState } from 'react';

type MenuPath = 'mobile' | 'profile';

export const useMenubarToggle = () => {
  const [profileOpen, setProfileOpen] = useState(false);
  const [mobileOpen, setMobileOpen] = useState(false);

  const setMenubar = (path: MenuPath) => {
    if (path === 'mobile') {
      setMobileOpen(prev => !prev);
```

```
      setProfileOpen(false);
    } else if (path === 'profile') {
      setProfileOpen(prev => !prev);
      setMobileOpen(false);
    }
  };

  return {
    profileOpen,
    mobileOpen,
    setMenubar,
  };
};
</file>

<file path="client/src/lib/hooks/Cookies/getdelete.ts">
'use server'
import { cookies } from "next/headers"
import type { CookieName } from "./setCookie"

export default async function getCookie(Name: CookieName,) {
  try {
    const cookieStore = await cookies()
    return cookieStore.get(Name)?.value
  } catch (error) {
    console.error(`Error setting cookie ${Name}:`, error);
    throw new Error("Authentication failed");
  }
}
</file>

<file path="client/src/lib/hooks/Cookies/setCookie.ts">
'use server'
import { cookies } from "next/headers"
const COOKIE_EXPIRY = 60 * 60 * 24 * 5;
export type CookieName = "userId" | "userRole"

export default async function setCookie(Name: CookieName, Value: string) {
  try {
    const cookieStore = await cookies()
    cookieStore.set(Name, Value, {
      httpOnly: true,
      secure: process.env.NODE_ENV === 'production',
      sameSite: 'lax',
      maxAge: COOKIE_EXPIRY,
      path: '/'
    })

    // Verify the cookie was set
    const verifyValue = cookieStore.get(Name)?.value;
    if (verifyValue !== Value) {
      console.error(`Failed to set cookie ${Name}. Expected: ${Value}, Got: $
{verifyValue}`);
      throw new Error("Failed to set cookie");
    }

    return true;
  } catch (error) {
    console.error(`Error setting cookie ${Name}:`, error);
    throw new Error("Authentication failed");
  }
}
</file>
```

<file path="client/src/lib/hooks/Cookies/setdelete.ts">
```
'use server'
import { cookies } from "next/headers"
import type { CookieName } from "./setCookie"

export default async function deleteCookie(Name: CookieName,) {
  try {
    const cookieStore = await cookies()
    cookieStore.delete(Name)

  } catch (error) {
    console.error(`Error setting cookie ${Name}:`, error);
    throw new Error("Authentication failed");
  }
}
```
</file>

<file path="client/src/lib/supabase/auth/logout.ts">
```
import { createClient } from '@/lib/supabase/client';

export const logout = async () => {
  const supabase = createClient();

  const { error } = await supabase.auth.signOut();

  if (error) {
    throw new Error(error.message);
  }
};
```
</file>

<file path="client/src/lib/supabase/auth/useAuth.ts">
```
'use client';

import { useEffect, useState } from 'react';
import { createClient } from '@/lib/supabase/client';
import type { User } from '@supabase/supabase-js';

export const useAuth = () => {
  const [user, setUser] = useState<User | null>(null);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    const supabase = createClient();

    // Get initial session
    const getSession = async () => {
      const {
        data: { session },
      } = await supabase.auth.getSession();
      setUser(session?.user ?? null);
      setLoading(false);
    };

    getSession();

    // Subscribe to auth changes
    const { data: authListener } = supabase.auth.onAuthStateChange((event,
session) => {
      setUser(session?.user ?? null);
      setLoading(false);
    });

    return () => {
```
</file>

```
      authListener?.subscription.unsubscribe();
    };
  }, []);

  return { user, loading };
};
</file>

<file path="client/src/lib/supabase/products/products.ts">
import { createClient } from '@/lib/supabase/client';

export interface Product {
  id: string;
  name: string;
  description: string;
  price: number;
  category: string;
  image_url: string;
  stock: number;
  created_at: string;
}

export const getProducts = async () => {
  const supabase = createClient();

  const { data, error } = await supabase.from('products').select('*');

  if (error) {
    console.error('Error fetching products:', error);
    return [];
  }

  return (data as Product[]) || [];
};

export const getProductsByCategory = async (category: string) => {
  const supabase = createClient();

  const { data, error } = await supabase
    .from('products')
    .select('*')
    .eq('category', category);

  if (error) {
    console.error('Error fetching products by category:', error);
    return [];
  }

  return (data as Product[]) || [];
};

export const getProductById = async (id: string) => {
  const supabase = createClient();

  const { data, error } = await supabase.from('products').select('*').eq('id',
id).single();

  if (error) {
    console.error('Error fetching product:', error);
    return null;
  }

  return (data as Product) || null;
};
```

</file>

<file path="client/src/lib/types/CarWashTypes.d.ts">
```
declare global {
  interface BookingFormData {
    service: string;
    addOns: string[];
    date: string;
    time: string;
    worker: string;
  }

  interface TimeSlotPickerProps {
    selectedDate: string;
    selectedService: string;
    onSelectTime: (time: string) => void;
  }

  interface CarWashBooking {
    id: string;
    userId: string;
    service: string;
    date: Date;
    time: string;
    worker: string;
    status: 'pending' | 'confirmed' | 'cancelled' | 'completed' | 'deleted';
    createdAt: Date;
    addOns?: string[];
    vehicleDetails?: string;
    isFree?: boolean;
  }

  type BookingStatus = 'pending' | 'loading' | 'success' | 'error';
}
export { };
```
</file>

<file path="client/src/lib/types/index.ts">
```
export interface Reward {
  name: string;
  pointsrequired: number;
}
export type paymentMethod = "instore" | "Cepitec transfer";

export interface yellowemption {
  name: string;
  date: string;
}
```
</file>

<file path="client/src/lib/types/OrderTypes.d.ts">
```
declare global {

  interface OrderDetails {
    userId: string;
    items: CartItem[];
    totalAmount: number;
    shippingAddress: string;
    paymentMethod: paymentMethod;
  }

  interface OrderItem {
    name: string;
    quantity: number;
```

```
  }

  type CartItem = ProductType & {
    quantity: number;
  };

  interface Order {
    orderId: string;
    userId?: string;
    orderDate: Date;
    products: CartItem[];
    quantity: number;
    totalAmount: number;
    status: 'pending' | 'processing' | 'completed' | 'cancelled';
    type: 'Takeaway' | 'Photo boot' | 'Laundry';
    location?: string;
    paymentMethod?: paymentMethod;
    deliveryAddress?: string;
    shippingAddress?: string;
  }
}
export { };
</file>

<file path="client/src/lib/utils.ts">
import { clsx, type ClassValue } from "clsx"
import { twMerge } from "tailwind-merge"

export function cn(...inputs: ClassValue[]) {
  return twMerge(clsx(inputs))
}
</file>

<file path="client/src/lib/utils/dateUtils.ts">
export const formatDate = (date: Date | string): string => {
  const dateObj = date instanceof Date ? date : new Date(date);
  return dateObj.toLocaleDateString('en-US', {
    year: 'numeric',
    month: 'long',
    day: 'numeric',
  });
};

export const generateTimeSlots = (start = 9, end = 17, interval = 60): string[]
=> {
  const slots: string[] = [];
  for (let hour = start; hour < end; hour++) {
    slots.push(
      `${hour.toString().padStart(2, '0')}:00`
    );
  }
  return slots;
};

export const isDateInPast = (date: Date | string): boolean => {
  const today = new Date();
  today.setHours(0, 0, 0, 0);
  const compayellowate = new Date(date);
  compayellowate.setHours(0, 0, 0, 0);
  return compayellowate < today;
};
</file>

<file path="client/supabase_migrations/add_profile_trigger.sql">
```

```
-- Create or replace the function with existence check
CREATE OR REPLACE FUNCTION public.handle_new_user()
RETURNS trigger
LANGUAGE plpgsql
SECURITY DEFINER
SET search_path = public
AS $$
BEGIN
  -- Only create a profile if one does not already exist for this id or uid
  IF NOT EXISTS (SELECT 1 FROM public.profiles WHERE id = NEW.id OR uid =
NEW.id::text) THEN
    INSERT INTO public.profiles (
      id,
      email,
      display_name,
      phone,
      role,
      uid,
      email_verified,
      photo_url,
      address,
      city,
      state,
      zip_code,
      country,
      theme,
      referral_code,
      preferences,
      saved_payment_methods,
      metadata,
      created_at,
      updated_at,
      last_login
    )
    VALUES (
      NEW.id,
      NEW.email,
      NEW.raw_user_meta_data ->> 'display_name',
      NEW.raw_user_meta_data ->> 'phone_number',
      'customer',
      NEW.id::text,
      COALESCE((NEW.email_confirmed_at IS NOT NULL), false),
      NULL,
      NULL,
      NULL,
      NULL,
      NULL,
      NULL,
      'system',
      NULL,
      jsonb_build_object(
        'dietaryRestrictions', '[]'::jsonb,
        'favoriteItems', '[]'::jsonb,
        'preferyellowPaymentMethod', 'credit_card',
        'communicationPreferences', jsonb_build_object(
          'email', true,
          'sms', false,
          'promotions', true
        )
      ),
      '[]'::jsonb,
      jsonb_build_object(
        'raw_user_meta_data', NEW.raw_user_meta_data,
        'app_metadata', NEW.raw_app_meta_data
```

```
      ),
      now(),
      now(),
      now()
    );
  END IF;

  RETURN NEW;
END;
$$;

-- Replace trigger: drop then create
DROP TRIGGER IF EXISTS on_auth_user_created ON auth.users;

CREATE TRIGGER on_auth_user_created
  AFTER INSERT ON auth.users
  FOR EACH ROW
  EXECUTE FUNCTION public.handle_new_user();
</file>

<file path="admin/src/components/forms/ImageUpload.tsx">
"use client";

import { useState, useRef } from "react";
import { HiPhoto, HiXMark } from "react-icons/hi2";
import Image from "next/image";
import { uploadImage } from "@/lib/supabase/storage";

type Props = {
    defaultValue?: string | null;
    onChange: (url: string, filePath?: string) => void;
    folder?: string;
    fieldName?: string;
};

export const ImageUpload = ({ defaultValue, onChange, folder = "uploads",
fieldName = "image_url" }: Props) => {
    const [preview, setPreview] = useState<string | null>(defaultValue || null);
    const [uploading, setUploading] = useState(false);
    const [value, setValue] = useState<string>(defaultValue || "");
    const [uploadedPath, setUploadedPath] = useState<string>(""); // Track for
cleanup
    const fileInputRef = useRef<HTMLInputElement>(null);

    const handleFileChange = async (e: React.ChangeEvent<HTMLInputElement>) => {
        const file = e.target.files?.[0];
        if (!file) return;

        // Create local preview
        const objectUrl = URL.createObjectURL(file);
        setPreview(objectUrl);
        setUploading(true);

        try {
            const publicUrl = await uploadImage(file, folder);
            const filePath = publicUrl.split(`${folder}/`)[1] || "";
            console.log("ImageUpload: Upload success, URL:", publicUrl);
            setValue(publicUrl);
            setUploadedPath(filePath);
            onChange(publicUrl, `${folder}/${filePath}`);
        } catch (error) {
            console.error("Upload failed:", error);
            alert("Failed to upload image. Please try again.");
            setPreview(defaultValue || null); // Revert on failure
```

```
            setValue(defaultValue || "");
        } finally {
            setUploading(false);
        }
    };

    const handleRemove = () => {
        console.log("ImageUpload: Removing image");
        setPreview(null);
        setValue("");
        setUploadedPath("");
        onChange("");
        if (fileInputRef.current) {
            fileInputRef.current.value = "";
        }
    };

    return (
        <div className="space-y-4">
            <div className="flex items-center gap-4">
                {preview ? (
                    <div className="relative h-40 w-40 overflow-hidden rounded-
lg border border-white/10 bg-yellow-900/50">
                        <Image
                            src={preview}
                            alt="Preview"
                            fill
                            className="object-cover"
                        />
                        <button
                            type="button"
                            onClick={handleRemove}
                            className="absolute right-2 top-2 rounded-full bg-
black/50 p-1 text-white hover:bg-rose-500 transition-colors"
                        >
                            <HiXMark className="h-4 w-4" />
                        </button>
                        {uploading && (
                            <div className="absolute inset-0 flex items-center
justify-center bg-black/50">
                                <div className="h-6 w-6 animate-spin rounded-
full border-2 border-indigo-500 border-t-transparent" />
                            </div>
                        )}
                    </div>
                ) : (
                    <button
                        type="button"
                        onClick={() => fileInputRef.current?.click()}
                        className="flex h-40 w-40 flex-col items-center justify-
center gap-2 rounded-lg border border-dashed border-white/20 bg-yellow-900/20
hover:bg-yellow-900/40 transition-colors"
                    >
                        <HiPhoto className="h-8 w-8 text-yellow-400" />
                        <span className="text-xs text-yellow-400">Upload
Image</span>
                    </button>
                )}
                <input
                    ref={fileInputRef}
                    type="file"
                    accept="image/*"
                    onChange={handleFileChange}
                    className="hidden"
```

```
              />
            </div>
            <input type="text" name={fieldName} value={value} readOnly
className="w-full text-xs bg-yellow-800 text-yellow-400 p-2 rounded" />
        </div>
    );
};
</file>

<file path="admin/src/components/layout/AdminTopbar.tsx">
import Link from "next/link";
import { HiOutlineBell, HiOutlineQuestionMarkCircle } from "react-icons/hi2";
import { logoutAction } from "@/lib/auth";
import type { AdminProfileSummary } from "@/lib/types";
import { NAV_ITEMS } from "./nav-items";

type Props = {
  profile: AdminProfileSummary;
};

export const AdminTopbar = ({ profile }: Props) => (
  <header className="flex flex-col gap-4 border-b border-white/5 bg-
yellow-950/60 px-4 py-4 backdrop-blur">
    <div>
      <p className="text-sm uppercase tracking-[0.35em] text-indigo-300">
        Operations
      </p>
      <h1 className="text-xl font-semibold text-white">
        Welcome back, {profile.display_name ?? "Admin"}
      </h1>
    </div>

    <div className="flex flex-col gap-3 md:flex-row md:items-center md:justify-
between">


      <div className="flex items-center gap-3">
        <Link
          href="/support"
          className="inline-flex items-center gap-2 rounded-full border border-
white/10 px-3 py-2 text-sm text-yellow-200 transition hover:border-indigo-400
hover:text-white"
        >
          <HiOutlineQuestionMarkCircle />
          Support
        </Link>
        <button
          type="button"
          className="rounded-full border border-white/10 p-2 text-yellow-200
transition hover:border-indigo-400 hover:text-white"
          aria-label="Notifications"
        >
          <HiOutlineBell className="text-lg" />
        </button>
        <form action={logoutAction}>
          <button
            type="submit"
            className="rounded-full border border-white/10 px-4 py-2 text-sm
text-yellow-200 transition hover:border-rose-500 hover:text-white"
          >
            Sign out
          </button>
        </form>
      </div>
```

```
    </div>
  </header>
);
</file>

<file path="admin/src/components/layout/SidebarNav.tsx">
"use client";

import Link from "next/link";
import { usePathname } from "next/navigation";
import {
  HiOutlineChartBar,
  HiOutlineClipboardDocumentList,
  HiOutlineCog,
  HiOutlineSquares2X2,
  HiOutlineEnvelope,
  HiOutlineUsers,
  HiOutlineCamera,
  HiOutlineStar,
  HiOutlineSparkles,
} from "react-icons/hi2";
import { PiBowlFoodBold } from "react-icons/pi";
import type { NavItem } from "./nav-items";

type Props = {
  items: NavItem[];
};

const iconMap: Record<string, React.ComponentType<{ className?: string }>> = {
  HiOutlineChartBar,
  PiBowlFoodBold,
  HiOutlineClipboardDocumentList,
  HiOutlineSquares2X2,
  HiOutlineEnvelope,
  HiOutlineUsers,
  HiOutlineCamera,
  HiOutlineStar,
  HiOutlineSparkles,
  HiOutlineCog,
};

export const SidebarNav = ({ items }: Props) => {
  const pathname = usePathname();

  return (
    <nav className="space-y-1">
      {items.map(({ label, href, icon, badge }) => {
        const isActive = pathname === href;
        const Icon = iconMap[icon];

        return (
          <Link
            key={href}
            href={href}
            className={`flex items-center justify-between rounded-xl px-4 py-3
text-sm font-medium transition ${isActive
              ? "bg-indigo-500/20 text-white"
              : "text-yellow-300 hover:bg-white/5 hover:text-white"
              }`}
          >
            <span className="flex items-center gap-3">
              {Icon && <Icon className="text-lg" />}
              {label}
            </span>
```

```
            {badge ? (
              <span className="rounded-full bg-white/10 px-2 py-0.5 text-[10px]
uppercase tracking-widest text-white">
                {badge}
              </span>
            ) : null}
          </Link>
        );
      })}
    </nav>
  );
};
</file>

<file path="admin/src/components/menu/products/AddProductForm.tsx">
"use client";

import { useFormState } from "react-dom";
import { SubmitButton } from "@/components/forms/SubmitButton";
import {
  createProductAction,
  type ProductActionState,
} from "@/lib/data/products-actions";
import { ImageUpload } from "@/components/forms/ImageUpload";

type Props = {
  categories: string[];
};

const initialState: ProductActionState = {};

export const AddProductForm = ({ categories }: Props) => {
  const [state, formAction] = useFormState(createProductAction, initialState);

  return (
    <form action={formAction} className="space-y-4 text-sm text-yellow-200">
      <div className="space-y-2">
        <label htmlFor="name">Product name</label>
        <input
          id="name"
          name="name"
          required
          placeholder="Mogodu Mondays"
          className="w-full rounded-lg border border-white/10 bg-yellow-900/60
px-3 py-2 text-white outline-none focus:border-indigo-400 focus:ring-2
focus:ring-indigo-400/40"
        />
      </div>

      <div className="space-y-2">
        <label htmlFor="category">Category</label>
        <input
          id="category"
          name="category"
          list="menu-categories"
          required
          placeholder="Warm bowls"
          className="w-full rounded-lg border border-white/10 bg-yellow-900/60
px-3 py-2 text-white outline-none focus:border-indigo-400 focus:ring-2
focus:ring-indigo-400/40"
        />
        <datalist id="menu-categories">
          {categories.map((category) => (
            <option key={category} value={category ?? ""} />
```

```
          ))}
        </datalist>
      </div>

      <div className="grid gap-3 md:grid-cols-2">
        <div className="space-y-2">
          <label htmlFor="price">Price (ZAR)</label>
          <input
            id="price"
            name="price"
            type="number"
            step="0.01"
            min="0"
            required
            placeholder="85"
            className="w-full rounded-lg border border-white/10 bg-yellow-900/60
px-3 py-2 text-white outline-none focus:border-indigo-400 focus:ring-2
focus:ring-indigo-400/40"
          />
        </div>
        <div className="space-y-2">
          <label htmlFor="stock">Stock</label>
          <input
            id="stock"
            name="stock"
            type="number"
            min="0"
            placeholder="25"
            className="w-full rounded-lg border border-white/10 bg-yellow-900/60
px-3 py-2 text-white outline-none focus:border-indigo-400 focus:ring-2
focus:ring-indigo-400/40"
          />
        </div>
      </div>

      <div className="space-y-2">
        <label htmlFor="badge">Badge (optional)</label>
        <input
          id="badge"
          name="badge"
          placeholder="New / Bestseller"
          className="w-full rounded-lg border border-white/10 bg-yellow-900/60
px-3 py-2 text-white outline-none focus:border-indigo-400 focus:ring-2
focus:ring-indigo-400/40"
        />
      </div>

      <div className="space-y-2">
        <label htmlFor="image_url">Product Image</label>
        <ImageUpload onChange={() => { }} folder="products" />
      </div>

      <div className="space-y-2">
        <label htmlFor="description">Story / description</label>
        <textarea
          id="description"
          name="description"
          rows={4}
          placeholder="Slow-cooked tripe with creamy samp, toasted chakalaka
crumbs, and herb oil."
          className="w-full rounded-lg border border-white/10 bg-yellow-900/60
px-3 py-2 text-white outline-none focus:border-indigo-400 focus:ring-2
focus:ring-indigo-400/40"
        />
```

```
      </div>

      {state.error ? (
        <p className="text-sm text-rose-400">{state.error}</p>
      ) : null}
      {state.success ? (
        <p className="text-sm text-emerald-400">{state.success}</p>
      ) : null}

      <SubmitButton label="Add to menu" loadingLabel="Saving..." />
    </form>
  );
};
</file>

<file path="admin/src/components/menu/products/ProductBoard.tsx">
"use client";

import { useMemo, useState } from "react";
import Link from "next/link";
import Image from "next/image";
import { HiOutlinePlus, HiOutlinePencil, HiChevronDown, HiChevronUp,
HiOutlineChatBubbleLeft } from "react-icons/hi2";
import type { ProductRecord, CommentRecord, UserFavoriteRecord } from
"@/lib/types";

type Props = {
  products: ProductRecord[];
  comments: CommentRecord[];
  favorites: UserFavoriteRecord[];
  onAddProduct?: () => void;
};

const normalize = (value?: string | null) => value?.toLowerCase() ?? "";

export const ProductBoard = ({ products, comments, favorites, onAddProduct }:
Props) => {
  const [search, setSearch] = useState("");
  const [category, setCategory] = useState("all");
  const [viewMode, setViewMode] = useState<"list" | "grid">("list");
  const [isExpanded, setIsExpanded] = useState(false);
  const [expandedComments, setExpandedComments] = useState<Set<string>>(new
Set());

  const categories = useMemo(() => {
    const unique = Array.from(
      new Set(products.map((product) => product.category_name ?? "")),
    ).filter(Boolean) as string[];
    return ["all", ...unique];
  }, [products]);

  const filtered = useMemo(() => {
    return products.filter((product) => {
      const matchesSearch =
        !search ||
        normalize(product.name).includes(search.toLowerCase()) ||
        normalize(product.description).includes(search.toLowerCase());
      const matchesCategory =
        category === "all" ||
        normalize(product.category_name) === category.toLowerCase();
      return matchesSearch && matchesCategory;
    });
  }, [category, products, search]);
```

```
  // Calculate product-specific data
  const productData = useMemo(() => {
    return filtered.map((product) => ({
      ...product,
      comments: comments.filter((c) => c.product_id === product.id),
    }));
  }, [filtered, comments]);

  const avgPrice =
    filtered.length > 0
      ? filtered.reduce((sum, product) => sum + (product.price ?? 0), 0) /
      filtered.length
      : 0;
  const lowStock = filtered.filter(
    (product) => (product.stock ?? 0) <= 10,
  ).length;
  const outOfStock = filtered.filter((product) => (product.stock ?? 0) ===
0).length;
  const totalStock = filtered.reduce((sum, product) => sum + (product.stock ??
0), 0);
  const totalValue = filtered.reduce((sum, product) => sum + (product.price ??
0) * (product.stock ?? 0), 0);

  // Favorites summary
  const productsWithFavorites = useMemo(() => {
    const favoritesByProduct = favorites.reduce((acc, fav) => {
      acc[fav.product_id] = (acc[fav.product_id] || 0) + 1;
      return acc;
    }, {} as Record<string, number>);

    return Object.entries(favoritesByProduct)
      .map(([productId, count]) => ({
        product: products.find((p) => p.id === productId),
        count,
      }))
      .filter((item) => item.product)
      .sort((a, b) => b.count - a.count)
      .slice(0, 5); // Top 5
  }, [products, favorites]);

  const toggleComments = (productId: string) => {
    setExpandedComments((prev) => {
      const newSet = new Set(prev);
      if (newSet.has(productId)) {
        newSet.delete(productId);
      } else {
        newSet.add(productId);
      }
      return newSet;
    });
  };

  return (
    <div className="rounded-2xl border border-white/10 bg-yellow-950/40 p-6">
      <div className="mb-4 flex items-center justify-between">
        <div className="flex-1">
          <h2 className="text-xl font-semibold text-white">Menu Management</h2>
          <p className="mt-1 text-sm text-yellow-400">
            Control availability, pricing, and storytelling for every PA
Catering experience
            {!isExpanded && ` â ¢ ${products.length} products`}
          </p>
        </div>
        <button
```

```
          type="button"
          onClick={() => setIsExpanded(!isExpanded)}
          className="flex items-center gap-2 rounded-lg bg-yellow-800 px-4 py-2
text-sm text-white hover:bg-yellow-700"
        >
          {isExpanded ? (
            <>
              <HiChevronUp className="h-4 w-4" />
              Hide
            </>
          ) : (
            <>
              <HiChevronDown className="h-4 w-4" />
              Show
            </>
          )}
        </button>
      </div>

      {isExpanded && (
        <div className="space-y-5">
          {/* Insights - Full Width */}
          <div className="space-y-4">
            <div className="rounded-xl border border-white/10 bg-yellow-900/50
p-4">
              <p className="mb-3 text-xs uppercase tracking-[0.3em] text-
yellow-500">
                Product Insights
              </p>
              <div className="grid grid-cols-2 gap-4 md:grid-cols-5">
                <div className="text-center">
                  <p className="text-xs text-yellow-400">Avg Price</p>
                  <p className="mt-1 text-lg font-semibold text-
white">R{avgPrice.toFixed(2)}</p>
                </div>
                <div className="text-center">
                  <p className="text-xs text-yellow-400">Total Stock</p>
                  <p className="mt-1 text-lg font-semibold text-
white">{totalStock}</p>
                </div>
                <div className="text-center">
                  <p className="text-xs text-yellow-400">Low Stock</p>
                  <p className="mt-1 text-lg font-semibold text-
yellow-400">{lowStock}</p>
                </div>
                <div className="text-center">
                  <p className="text-xs text-yellow-400">Out of Stock</p>
                  <p className="mt-1 text-lg font-semibold text-
rose-400">{outOfStock}</p>
                </div>
                <div className="text-center">
                  <p className="text-xs text-yellow-400">Total Value</p>
                  <p className="mt-1 text-lg font-semibold text-
emerald-400">R{totalValue.toFixed(2)}</p>
                </div>
              </div>
            </div>

            {/* Favorites Summary */}
            {productsWithFavorites.length > 0 && (
              <div className="rounded-xl border border-white/10 bg-yellow-900/50
p-4">
                <p className="mb-3 text-xs uppercase tracking-[0.3em] text-
yellow-500">
```

```
                Top Favorited Products
              </p>
              <div className="space-y-2">
                {productsWithFavorites.map(({ product, count }) => (
                  <div
                    key={product!.id}
                    className="flex items-center justify-between rounded-lg
bg-yellow-800/40 px-3 py-2"
                  >
                    <span className="text-sm
text-white">{product!.name}</span>
                    <span className="text-xs text-emerald-400">{count}
favorites</span>
                  </div>
                ))}
              </div>
            </div>
          )}
        </div>

        {/* Search and Category Filters */}
        <div className="grid gap-3 md:grid-cols-2">
          <label className="space-y-2 text-sm text-yellow-300">
            <span>Search</span>
            <input
              type="search"
              placeholder="Name or description"
              value={search}
              onChange={(event) => setSearch(event.target.value)}
              className="w-full rounded-xl border border-white/10 bg-
yellow-900/60 px-3 py-2 text-white outline-none focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
            />
          </label>

          <label className="space-y-2 text-sm text-yellow-300">
            <span>Category</span>
            <select
              value={category}
              onChange={(event) => setCategory(event.target.value)}
              className="w-full rounded-xl border border-white/10 bg-
yellow-900/60 px-3 py-2 text-white outline-none focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
            >
              {categories.map((item) => (
                <option key={item} value={item}>
                  {item === "all" ? "All menu sections" : item}
                </option>
              ))}
            </select>
          </label>
        </div>

        <div className="flex items-center justify-between">
          <div className="flex gap-2">
            <button
              type="button"
              onClick={() => setViewMode("list")}
              className={`rounded-lg px-3 py-1.5 text-sm ${viewMode === "list"
                ? "bg-indigo-600 text-white"
                : "bg-yellow-800 text-yellow-400 hover:text-white"
                }`}
            >
              List
```

```
              </button>
              <button
                type="button"
                onClick={() => setViewMode("grid")}
                className={`rounded-lg px-3 py-1.5 text-sm ${viewMode === "grid"
                  ? "bg-indigo-600 text-white"
                  : "bg-yellow-800 text-yellow-400 hover:text-white"
                  }`}
              >
                Grid
              </button>
            </div>

            {onAddProduct && (
              <button
                type="button"
                onClick={onAddProduct}
                className="flex items-center gap-2 rounded-full bg-indigo-600
px-4 py-2 text-sm font-medium text-white hover:bg-indigo-700"
              >
                <HiOutlinePlus className="h-5 w-5" />
                Add Product
              </button>
            )}
          </div>

          <div className={viewMode === "grid" ? "grid gap-4 md:grid-cols-2
lg:grid-cols-3" : "grid gap-4 md:grid-cols-2"}>
            {productData.length === 0 ? (
              <div className="col-span-full rounded-2xl border border-white/10
bg-yellow-900/40 p-8 text-center text-yellow-500">
                No menu items match your filters.
              </div>
            ) : (
              productData.map((product) => (
                <article
                  key={product.id}
                  className="rounded-2xl border border-white/10 bg-yellow-900/40
p-4 shadow-inner shadow-black/30"
                >
                  {product.image_url && (
                    <div className="relative mb-4 aspect-[4/3] w-full overflow-
hidden rounded-md bg-black/20">
                      <Image
                        src={product.image_url}
                        alt={product.name}
                        fill
                        className="rounded-md object-contain transition-
transform duration-300 hover:scale-110"
                      />
                    </div>
                  )}

                  <div className="flex items-start justify-between gap-4">
                    <div className="flex-1">
                      <p className="text-xs uppercase tracking-[0.3em] text-
yellow-500">
                        {product.category_name ?? "Uncategorised"}
                      </p>
                      <h3 className="text-lg font-semibold text-white flex
items-center gap-2">
                        {product.name}
                        {product.is_hidden && (
                          <span className="rounded-full border border-
```

```
yellow-500/40 bg-yellow-500/10 px-2 py-0.5 text-[10px] uppercase tracking-wider
text-yellow-500">
                          Hidden
                        </span>
                      )}
                    </h3>
                    <p className="text-sm text-yellow-400">
                      {product.description ?? "No description yet."}
                    </p>
                  </div>
                  <div className="text-right">
                    <p className="text-2xl font-semibold text-white">
                      R{product.price?.toFixed(2) ?? "0.00"}
                    </p>
                    <p className="text-xs text-yellow-500">
                      {product.stock ?? 0} in stock
                    </p>
                    <Link
                      href={`/menu/products/${product.id}/edit`}
                      className="mt-2 flex items-center gap-1 rounded-lg bg-
yellow-800 px-3 py-1.5 text-xs text-white hover:bg-yellow-700"
                    >
                      <HiOutlinePencil className="h-3 w-3" />
                      Edit
                    </Link>
                  </div>
                </div>

                {product.badge && (
                  <span className="mt-2 inline-block rounded-full border
border-indigo-400/40 px-3 py-1 text-xs uppercase tracking-[0.3em] text-
indigo-200">
                    {product.badge}
                  </span>
                )}

                {/* Product Stats */}
                <div className="mt-3 flex items-center gap-4 text-xs text-
yellow-500">
                  <span>{product.likes?.length ?? 0} likes</span>
                  <button
                    type="button"
                    onClick={() => toggleComments(product.id)}
                    className="flex items-center gap-1 hover:text-yellow-300"
                  >
                    <HiOutlineChatBubbleLeft className="h-3 w-3" />
                    {product.comments.length} comments
                  </button>
                </div>

                {/* Comments Section */}
                {expandedComments.has(product.id) && (
                  <div className="mt-3 rounded-lg border border-white/10 bg-
yellow-800/40 p-3">
                    <h4 className="mb-2 text-xs font-semibold uppercase
tracking-wider text-yellow-400">
                      Comments
                    </h4>
                    {product.comments.length === 0 ? (
                      <p className="text-xs text-yellow-500">No comments
yet</p>
                    ) : (
                      <div className="space-y-2">
                        {product.comments.map((comment) => (
```

```
                            <div
                              key={comment.id}
                              className="rounded bg-yellow-900/60 p-2"
                            >
                              <p className="text-xs font-semibold text-white">
                                {comment.user_name || "Anonymous"}
                              </p>
                              <p className="mt-1 text-xs text-yellow-300">
                                {comment.body}
                              </p>
                              <p className="mt-1 text-xs text-yellow-600">
                                {new
Date(comment.created_at).toLocaleDateString()}
                              </p>
                            </div>
                          ))}
                        </div>
                      )}
                    </div>
                  )}
                </article>
              ))
            )}
          </div>
        </div>
      )}
    </div>
  );
};
</file>

<file path="admin/src/components/menu/QuickEditForm.tsx">
"use client";

import { useFormState } from "react-dom";
import { SubmitButton } from "@/components/forms/SubmitButton";
import {
  type ProductActionState,
  updateProductAction,
} from "@/lib/data/products-actions";
import type { ProductRecord } from "@/lib/types";

type Props = {
  product: ProductRecord;
  categories?: string[];
};

const initialState: ProductActionState = {};

export const QuickEditForm = ({ product, categories = [] }: Props) => {
  const [state, formAction] = useFormState(updateProductAction, initialState);

  return (
    <form action={formAction} className="space-y-3 text-sm text-yellow-200">
      <input type="hidden" name="id" value={product.id} />

      <label className="space-y-2 block">
        <span>Product Name</span>
        <input
          name="name"
          type="text"
          defaultValue={product.name}
          className="w-full rounded-lg border border-white/10 bg-yellow-900/60
px-3 py-2 text-white outline-none focus:border-indigo-400 focus:ring-2
```

```
focus:ring-indigo-400/40"
        />
      </label>

      <label className="space-y-2 block">
        <span>Category</span>
        {categories.length > 0 ? (
          <select
            name="category"
            defaultValue={product.category_name ?? ""}
            className="w-full rounded-lg border border-white/10 bg-yellow-900/60
px-3 py-2 text-white outline-none focus:border-indigo-400 focus:ring-2
focus:ring-indigo-400/40"
          >
            {categories.filter(cat => cat !== "all").map((cat) => (
              <option key={cat} value={cat}>
                {cat}
              </option>
            ))}
          </select>
        ) : (
          <input
            name="category"
            type="text"
            defaultValue={product.category_name ?? ""}
            className="w-full rounded-lg border border-white/10 bg-yellow-900/60
px-3 py-2 text-white outline-none focus:border-indigo-400 focus:ring-2
focus:ring-indigo-400/40"
          />
        )}
      </label>

      <div className="grid gap-3 md:grid-cols-2">
        <label className="space-y-2">
          <span>Price (ZAR)</span>
          <input
            name="price"
            type="number"
            step="0.01"
            min="0"
            defaultValue={product.price ?? 0}
            className="w-full rounded-lg border border-white/10 bg-yellow-900/60
px-3 py-2 text-white outline-none focus:border-indigo-400 focus:ring-2
focus:ring-indigo-400/40"
          />
        </label>
        <label className="space-y-2">
          <span>Stock</span>
          <input
            name="stock"
            type="number"
            min="0"
            defaultValue={product.stock ?? 0}
            className="w-full rounded-lg border border-white/10 bg-yellow-900/60
px-3 py-2 text-white outline-none focus:border-indigo-400 focus:ring-2
focus:ring-indigo-400/40"
          />
        </label>
      </div>

      <label className="space-y-2 block">
        <span>Image URL</span>
        <input
          name="image_url"
```

```
          type="url"
          defaultValue={product.image_url ?? ""}
          placeholder="https://example.com/image.jpg"
          className="w-full rounded-lg border border-white/10 bg-yellow-900/60
px-3 py-2 text-white outline-none focus:border-indigo-400 focus:ring-2
focus:ring-indigo-400/40"
        />
      </label>

      <label className="space-y-2 block">
        <span>Badge</span>
        <input
          name="badge"
          defaultValue={product.badge ?? ""}
          placeholder="Chef's pick"
          className="w-full rounded-lg border border-white/10 bg-yellow-900/60
px-3 py-2 text-white outline-none focus:border-indigo-400 focus:ring-2
focus:ring-indigo-400/40"
        />
      </label>

      <label className="space-y-2 block">
        <span>Description</span>
        <textarea
          name="description"
          rows={3}
          defaultValue={product.description ?? ""}
          className="w-full rounded-lg border border-white/10 bg-yellow-900/60
px-3 py-2 text-white outline-none focus:border-indigo-400 focus:ring-2
focus:ring-indigo-400/40"
        />
      </label>

      {state.error ? (
        <p className="text-sm text-rose-400">{state.error}</p>
      ) : null}
      {state.success ? (
        <p className="text-sm text-emerald-400">{state.success}</p>
      ) : null}

      <SubmitButton
        label="Save changes"
        loadingLabel="Updating..."
        className="bg-yellow-800"
      />
    </form>
  );
};
</file>

<file path="admin/src/components/profiles/ProfileDetailModal.tsx">
"use client";

import { useState } from "react";
import { HiXMark, HiOutlineShoppingBag, HiOutlineHeart } from "react-icons/hi2";
import type { ProfileRecord, SupabaseOrderRecord } from "@/lib/types";
import { formatDistanceToNow } from "date-fns";

type Props = {
    profile: ProfileRecord;
    orders: SupabaseOrderRecord[];
    favorites: any[];
    onClose: () => void;
};
```

```
export const ProfileDetailModal = ({ profile, orders, favorites, onClose }:
Props) => {
    const [activeTab, setActiveTab] = useState<"info" | "orders" |
"favorites">("info");

    return (
        <div className="fixed inset-0 z-50 flex items-center justify-center bg-
black/60 p-4 backdrop-blur-sm">
            <div className="w-full max-w-4xl max-h-[90vh] overflow-y-auto
rounded-2xl border border-white/10 bg-yellow-900/95 p-6">
                {/* Header */}
                <div className="mb-6 flex items-start justify-between">
                    <div>
                        <h2 className="text-2xl font-bold text-white">
                            {profile.display_name || "No name"}
                        </h2>
                        <p className="text-sm text-yellow-400">{profile.email}
</p>
                        <div className="mt-2 flex gap-2">
                            <span className="rounded-full bg-indigo-500/20 px-2
py-0.5 text-xs font-medium uppercase text-indigo-300">
                                {profile.role}
                            </span>
                            <span className="rounded-full bg-purple-500/20 px-2
py-0.5 text-xs font-medium uppercase text-purple-300">
                                {profile.tier_status}
                            </span>

                        </div>
                    </div>
                    <button
                        type="button"
                        onClick={onClose}
                        className="rounded-lg bg-yellow-800 p-2 text-white
hover:bg-yellow-700"
                    >
                        <HiXMark className="h-5 w-5" />
                    </button>
                </div>

                {/* Tabs */}
                <div className="mb-6 flex gap-2 border-b border-white/10">
                    <button
                        type="button"
                        onClick={() => setActiveTab("info")}
                        className={`px-4 py-2 text-sm font-medium transition-
colors ${activeTab === "info"
                            ? "border-b-2 border-indigo-500 text-indigo-400"
                            : "text-yellow-400 hover:text-white"
                            }`}
                    >
                        Profile Info
                    </button>
                    <button
                        type="button"
                        onClick={() => setActiveTab("orders")}
                        className={`flex items-center gap-2 px-4 py-2 text-sm
font-medium transition-colors ${activeTab === "orders"
                            ? "border-b-2 border-indigo-500 text-indigo-400"
                            : "text-yellow-400 hover:text-white"
                            }`}
                    >
                        <HiOutlineShoppingBag className="h-4 w-4" />
```

```
                        Orders ({orders.length})
                    </button>
                    <button
                        type="button"
                        onClick={() => setActiveTab("favorites")}
                        className={`flex items-center gap-2 px-4 py-2 text-sm
font-medium transition-colors ${activeTab === "favorites"
                            ? "border-b-2 border-indigo-500 text-indigo-400"
                            : "text-yellow-400 hover:text-white"
                            }`}
                    >
                        <HiOutlineHeart className="h-4 w-4" />
                        Favorites ({favorites.length})
                    </button>
                </div>

                {/* Content */}
                <div className="space-y-4">
                    {activeTab === "info" && (
                        <div className="grid gap-4 md:grid-cols-2">
                            <InfoField label="User ID" value={profile.uid ||
"-"} />
                            <InfoField label="Phone" value={profile.phone ||
"-"} />
                            <InfoField label="Address" value={profile.address ||
"-"} />
                            <InfoField label="City" value={profile.city ||
"-"} />
                            <InfoField label="State" value={profile.state ||
"-"} />
                            <InfoField label="Zip Code" value={profile.zip_code
|| "-"} />
                            <InfoField label="Country" value={profile.country ||
"-"} />
                            <InfoField
                                label="Tier Status"
                                value={profile.tier_status || "Bronze"}
                            />

                            <InfoField
                                label="Last Login"
                                value={profile.last_login
                                    ? formatDistanceToNow(new
Date(profile.last_login), { addSuffix: true })
                                    : "Never"
                                }
                            />
                            <InfoField
                                label="Account Created"
                                value={formatDistanceToNow(new
Date(profile.created_at), { addSuffix: true })}
                            />
                        </div>
                    )}

                    {activeTab === "orders" && (
                        <div className="space-y-3">
                            {orders.length === 0 ? (
                                <p className="text-center text-sm text-
yellow-500">No orders yet</p>
                            ) : (
                                orders.map((order) => (
                                    <div
                                        key={order.id}
```

```
                                        className="rounded-lg border border-
white/10 bg-yellow-800/40 p-4"
                                    >
                                        <div className="flex items-start
justify-between">
                                            <div>
                                                <p className="font-semibold
text-white">
                                                    Order #{order.id.slice(0,
8)}
                                                </p>
                                                <p className="text-sm text-
yellow-400">
                                                    {formatDistanceToNow(new
Date(order.created_at), {
                                                        addSuffix: true,
                                                    })}
                                                </p>
                                            </div>
                                            <span className="rounded-full bg-
emerald-500/20 px-2 py-0.5 text-xs font-medium uppercase text-emerald-300">
                                                {order.status}
                                            </span>
                                        </div>
                                        <div className="mt-2 flex justify-
between text-sm">
                                            <span className="text-
yellow-400">Total:</span>
                                            <span className="font-semibold text-
white">
                                                ${order.total_price?.toFixed(2)
|| "0.00"}
                                            </span>
                                        </div>
                                    </div>
                                ))
                            )}
                        </div>
                    )}

                    {activeTab === "favorites" && (
                        <div className="grid gap-3 md:grid-cols-2">
                            {favorites.length === 0 ? (
                                <p className="col-span-2 text-center text-sm
text-yellow-500">No favorites yet</p>
                            ) : (
                                favorites.map((fav: any) => (
                                    <div
                                        key={fav.product_id}
                                        className="rounded-lg border border-
white/10 bg-yellow-800/40 p-3"
                                    >
                                        <div className="flex items-center
gap-3">
                                            {fav.products?.image_url && (
                                                <img
                                                    src={fav.products.image_url}
                                                    alt={fav.products.name}
                                                    className="h-16 w-16
rounded-lg object-cover"
                                                />
                                            )}
                                            <div className="flex-1">
                                                <p className="font-semibold
```

```
text-white">
                                                    {fav.products?.name ||
"Product"}
                                                </p>
                                                <p className="text-sm text-
yellow-400">
                                                    $
{fav.products?.price?.toFixed(2) || "0.00"}
                                                </p>
                                            </div>
                                        </div>
                                    </div>
                                ))
                            )}
                        </div>
                    )}
                </div>
            </div>
        </div>
    );
};

type InfoFieldProps = {
    label: string;
    value: string;
};

const InfoField = ({ label, value }: InfoFieldProps) => (
    <div className="rounded-lg border border-white/10 bg-yellow-800/40 p-3">
        <p className="text-xs text-yellow-500">{label}</p>
        <p className="mt-1 text-sm font-medium text-white">{value}</p>
    </div>
);
</file>

<file path="admin/src/lib/data/dashboard.ts">
import { createSupabaseServerClient } from "@/lib/supabase/server";
import type { ProductRecord, SupabaseOrderRecord } from "@/lib/types";

export type RevenuePoint = {
  date: string;
  total: number;
};

export type DashboardStats = {
  totalOrders: number;
  pendingOrders: number;
  fulfillmentRate: number;
  revenue: number;
  productCount: number;
  lowStockCount: number;
};

export type DashboardData = {
  stats: DashboardStats;
  recentOrders: SupabaseOrderRecord[];
  lowInventory: ProductRecord[];
  revenueSeries: RevenuePoint[];
};

const asCurrency = (value: number) =>
  Math.round((value + Number.EPSILON) * 100) / 100;

const buildRevenueSeries = (orders: SupabaseOrderRecord[]): RevenuePoint[] => {
```

```
  const buckets = new Map<string, number>();

  orders.forEach((order) => {
    const dateKey = new Date(order.created_at).toISOString().slice(0, 10);
    const total = buckets.get(dateKey) ?? 0;
    buckets.set(dateKey, total + (order.total_price ?? 0));
  });

  return Array.from(buckets.entries())
    .sort(([a], [b]) => (a > b ? 1 : -1))
    .slice(-7)
    .map(([date, total]) => ({
      date,
      total: asCurrency(total),
    }));
};

export const fetchDashboardData = async (): Promise<DashboardData> => {
  const supabase = await createSupabaseServerClient();

  const [
    { data: ordersRaw, error: ordersError },
    { data: productsRaw, count: productCount, error: productsError },
  ] = await Promise.all([
    supabase
      .from("orders")
      .select(

"id,user_id,items,total_price,total_quantity,status,created_at,updated_at",
      )
      .order("created_at", { ascending: false })
      .limit(50),
    supabase
      .from("products")
      .select("id,name,category_name,price,stock,image_url,badge", {
        count: "exact",
      }),
  ]);

  if (ordersError) {
    console.error("Failed to fetch orders for dashboard", ordersError);
  }

  if (productsError) {
    console.error("Failed to fetch products for dashboard", productsError);
  }

  const orders = (ordersRaw ?? []) as unknown as SupabaseOrderRecord[];
  const products = (productsRaw ?? []) as ProductRecord[];

  const totalOrders = orders.length;
  const revenue = asCurrency(
    orders.reduce((sum, order) => sum + (order.total_price ?? 0), 0),
  );
  const pendingOrders = orders.filter(
    (order) => order.status === "pending",
  ).length;
  const completedOrders = orders.filter(
    (order) => order.status === "completed",
  ).length;
  const fulfillmentRate = totalOrders
    ? Math.round((completedOrders / totalOrders) * 100)
    : 0;
```

```
  const lowInventory = products
    .filter((product) => (product.stock ?? 0) <= 10)
    .slice(0, 5);

  return {
    stats: {
      totalOrders,
      revenue,
      pendingOrders,
      fulfillmentRate,
      productCount: productCount ?? products.length,
      lowStockCount: lowInventory.length,
    },
    recentOrders: orders.slice(0, 6),
    lowInventory,
    revenueSeries: buildRevenueSeries(orders),
  };
};
</file>

<file path="admin/src/lib/types/database-records.ts">
// Type definitions for database records used across admin interface

export interface CommentRecord {
    id: string;
    product_id: string | null;
    user_id: string | null;
    user_name: string | null;
    body: string;
    created_at: string;
}

export interface ContactRecord {
    id: string;
    name: string;
    email: string;
    phone: string | null;
    message: string;
    created_at: string;
}

export interface PhotoBookingRecord {
    id: string;
    name: string;
    email: string;
    phone: string | null;
    date: string;
    time: string;
    package: string;
    people: number;
    message: string | null;
    created_at: string;
}

export interface CategoryRecord {
    id: string;
    category_name: string;
    image: string | null;
    description: string | null;
    created_at: string;
    is_hidden?: boolean;
}

export interface ProfileRecord {
```

```
    id: string;
    email: string | null;
    display_name: string | null;
    phone: string | null;
    metadata: Record<string, unknown> | null;
    created_at: string;
    role: string;
    uid: string | null;
    email_verified: boolean;
    photo_url: string | null;
    address: string | null;
    city: string | null;
    state: string | null;
    zip_code: string | null;
    country: string | null;
    theme: string;
    tier_status: string;
    referral_code: string | null;
    preferences: Record<string, unknown>;
    saved_payment_methods: unknown[];
    updated_at: string;
    last_login: string;
}

export interface UserFavoriteRecord {
    user_id: string;
    product_id: string;
    created_at: string;
}
</file>

<file path="admin/src/lib/types/users.ts">
import type { PaymentMethod, Reward, Yellowemption } from "./base";
import type { Order } from "./orders";

export type UserRole =
  | "coFounder"
  | "Ceo"
  | "Manager"
  | "kitchenStaff"
  | "Cashier"
  | "Customer"
  | "admin";

export interface UserPreferences {
  dietaryRestrictions: string[];
  favoriteItems: string[];
  preferyellowPaymentMethod: PaymentMethod;
  communicationPreferences: {
    email: boolean;
    sms: boolean;
    promotions: boolean;
  };
}

export interface UserProfile {
  uid: string;
  displayName: string | null;
  email: string | null;
  emailVerified: boolean;
  photoURL: string | null;
  phoneNumber: string | null;
  role: UserRole;
  address: string;
```

```
  city: string;
  state: string;
  zipCode: string;
  country: string;
  theme: "system" | "light" | "dark";
  tierStatus: string;
  referralCode: string;
  preferences: UserPreferences;
  savedPaymentMethods?: PaymentMethod[];
  createdAt: Date;
  updatedAt: Date;
  lastLogin?: Date;
}

export interface AdminProfileSummary {
  id: string;
  email?: string;
  display_name?: string;
  role: UserRole;
}
</file>

<file path="client/src/app/about/loading.tsx">
import Loading from '@/components/ui/Loading';

export default function LoadingPage() {
      return <Loading message='Loading About PA Luxe Creation...' />;
}
</file>

<file path="client/src/app/blog/page.tsx">
import React from 'react';
import BlogPostList from '@/app/blog/components/BlogPostList';
import Main from '@/components/ui/layout/Main';
import { FaBlog } from 'react-icons/fa';
import Section from '@/components/ui/layout/Section';

const BlogNewsPage: React.FC = () => {
      return (
            <Main
                  tittle='Blog'
                  Icon={FaBlog}>
                  <Section>
                        <BlogPostList />
                  </Section>
            </Main>
      );
};

export default BlogNewsPage;
</file>

<file path="client/src/app/contact/components/ContactInfo.tsx">
import Section from '@/components/ui/layout/Section';
import { IoLocationSharp, IoCall, IoMail } from 'react-icons/io5';

const ContactInfo: React.FC = () => {
      return (
            <Section tittle='Contact Information' >
                  <div className='md:flex md:flex-wrap md:gap-4'>

                        <article className='flex items-center  gap-2 p-4 '>

                              <IoLocationSharp className='text-white w-10 h-10
```

```
' />
                              <p>
                                    <strong>Address:</strong> 123 Main Street,
Phalaborwa, 1390
                              </p>
                        </article>
                        <article className='flex items-center  gap-2 p-2'>
                              <IoCall className='text-white w-10 h-10  ' />
                              <p>
                                    <strong>Phone Number:</strong> (+27) 15 781
1234
                              </p>
                        </article>
                        <article className='flex items-center  gap-2 p-2'>
                              <IoMail className='text-white w-10 h-10  ' />
                              <p>
                                    <strong>Email Address:</strong>
contact@centraleatery.co.za
                              </p>
                        </article>
                  </div>
            </Section>
      );
};

export default ContactInfo;
</file>

<file path="client/src/app/contact/components/MapEmbed.tsx">
const MapEmbed = () => (
  <div className="w-full h-96 rounded-lg overflow-hidden mt-8">
    <iframe
      src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!
1d387193.3059669869!2d-74.25986773715416!3d40.69767006372085!2m3!1f0!2f0!3f0!
3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x89c24fa5d33f083b%3A0xc80b8f06e177fe62!
2sNew%20York%2C%20NY%2C%20USA!5e0!3m2!1sen!2s!4v1234567890"
      width="100%"
      height="100%"
      style={{ border: 0 }}
      allowFullScreen
      loading="lazy"
      referrerPolicy="no-referrer-when-downgrade"
      className="rounded-lg"
    />
  </div>
);

export default MapEmbed;
</file>

<file path="client/src/app/gallery/page.tsx">
import ImageGallery from '@/app/gallery/components/ImageGallery';
import Main from '@/components/ui/layout/Main';
import { FcGallery } from 'react-icons/fc';

export default function Gallery() {
      return (
            <Main
                  tittle='Gallery'
                  Icon={FcGallery}
                  heading='Find Your Convenience'>
                  <ImageGallery />
            </Main>
      );
```

```
}
</file>

<file path="client/src/app/home-components/HeroSection.tsx">
import Image from 'next/image';
import Link from '@/components/ui/Link';
import { FaShoppingCart } from 'react-icons/fa';
import Section from '@/components/ui/layout/Section';
import { BiCamera } from 'react-icons/bi';
const HeroSection: React.FC = () => {
      return (
            <Section
                  tittle='Ever bite is a delight, ever wash is a luxury'
                  className='flex flex-col items-center text-center'>
                  <Image
                        src='/PA_Logo.png'
                        alt='PA Luxe Creation Logo'
                        width={200}
                        height={200}
                        priority
                  />
                  <p className='mt-4 text-lg text-shadow-sm text-shadow-black/50
font-bold max-w-2xl'>
                        Experience the unique convenience of delicious food and
a state-of-the-art Photo boot, all in
                        one place in Evander.
                  </p>
                  <div className='flex bg-transparent w-full justify-center
gap-6 mt-8'>
                        <Link
                              variant='button'
                              href='/menu'
                              className='flex items-center gap-2'>
                              <FaShoppingCart />
                              <span>Order Food</span>
                        </Link>
                        <Link
                              variant='button'
                              href='/photo'
                              className='flex items-center gap-2'>
                              <BiCamera />
                              <span>Book 360 phot boot</span>
                        </Link>
                  </div>
            </Section>
      );
};

export default HeroSection;
</file>

<file path="client/src/app/layout.css">
@import 'tailwindcss';
@import './globals.css';

@layer base {
      body {
            @apply m-0 p-0 text-white bg-black  bg-fixed h-full relative;
            font-family: ui-sans-serif, system-ui, -apple-system,
BlinkMacSystemFont, 'Segoe UI', Roboto,
                  'Helvetica Neue', Arial, 'Noto Sans', sans-serif, 'Apple Color
Emoji', 'Segoe UI Emoji',
                  'Segoe UI Symbol', 'Noto Color Emoji';
            background-image: url('/BackGround.png');
```

```
            background-size: cover;
            background-repeat: no-repeat;
            color: white;
      }

      article {
            @apply bg-yellow-700 rounded-md py-2 md:p-6 my-3 mt-6 mb-6 px-8
shadow-sm shadow-black;
      }
      nav {
            @apply m-0 p-0 md:p-6;
      }
      footer {
            @apply w-full bg-white/10 border border-yellow-500 rounded-md p-6
my-3 mt-6 mb-6 px-8;
      }
}
</file>

<file path="client/src/app/menu/[Products]/[product]/components/AddtoCart.tsx">
'use client';

import { useCart } from '@/lib/context/CartContext';
import { useToast } from '@/lib/context/CartContext';
import Button from '@/components/ui/Button';
import { HiShoppingCart } from 'react-icons/hi2';
import { useRouter } from 'next/navigation';

interface AddtoCartProps extends React.ButtonHTMLAttributes<HTMLButtonElement> {
      product: ProductType;
}

const AddtoCart: React.FC<AddtoCartProps> = ({ product, ...props }) => {
      const { addItem } = useCart();
      const { showToast } = useToast();
      const router = useRouter();

      return (
            <Button
                  variant='primary'
                  className='w-full bg-gradient-to-r from-amber-600 to-
yellow-600 hover:from-amber-500 hover:to-yellow-500 border-0 shadow-lg
hover:shadow-amber-500/50 transition-all duration-200 font-semibold'
                  size='sm'
                  onClick={() => {
                        addItem(product);
                        showToast(`Added ${product.Name} (R$
{product.Price.toFixed(2)}) to cart!`);
                        router.push('/menu/cart');
                  }}
                  {...props}>
                  <HiShoppingCart
                        color='white'
                        size={20}
                  />
                  Add to Cart
            </Button>
      );
};

export default AddtoCart;
</file>

<file
```

```
path="client/src/app/menu/[Products]/[product]/components/ProductInfo.tsx">
import React from 'react';
import { FaExclamationTriangle, FaLeaf, FaFire } from 'react-icons/fa';

interface ProductInfoProps {
      product: ProductType;
}

const ProductInfo: React.FC<ProductInfoProps> = ({ product }) => {
      // Mock nutritional data - can be replaced with real data from product
      const nutritionalInfo = {
            calories: product?.nutritional?.calories || 450,
            protein: product?.nutritional?.protein || '12g',
            carbs: product?.nutritional?.carbs || '45g',
            fat: product?.nutritional?.fat || '18g',
            fiber: product?.nutritional?.fiber || '3g',
            sugar: product?.nutritional?.sugar || '8g',
      };

      // Mock allergen info - can be replaced with real data from product
      const allergens = product?.allergens || ['Gluten', 'Dairy', 'Eggs'];
      const isVegetarian = product?.isVegetarian || true;
      const isSpicy = product?.isSpicy || true;

      return (
            <div className=' p-2 rounded-md text-white space-y-8'>
                  {/* Product Name and Price */}
                  <article>
                        <h1 className='text-4xl font-extrabold'>{product.Name}
</h1>
                        <p className='text-3xl text-center text-yellow-500 font-
bold text-white mt-2'>
                              R{product.Price.toFixed(2)}
                        </p>
                  </article>

                  {/* Description & Dietary Tags */}
                  <article>
                        <h2 className='text-2xl font-bold text-white
mb-3'>Description</h2>
                        <ul className='list-disc list-inside space-y-1.5 text-
white'>
                              {product.Description.map((d: string, index:
number) => (
                                    <li key={index}>{d}</li>
                              ))}
                        </ul>
                        <div className='flex flex-wrap gap-3 mt-4'>
                              {isVegetarian && (
                                    <span className='flex items-center gap-2
px-3 py-1 bg-yellow-600 text-white rounded-full text-sm font-semibold'>
                                          <FaLeaf /> Vegetarian
                                    </span>
                              )}
                              {isSpicy && (
                                    <span className='flex items-center gap-2
px-3 py-1 bg-yellow-600 text-white rounded-full text-sm font-semibold'>
                                          <FaFire /> Spicy
                                    </span>
                              )}
                        </div>
                  </article>

                  {/* Allergen Warnings Card */}
```

```
                  {allergens.length > 0 && (
                        <article className='bg-yellow-500/20 border border-
yellow-500/50 rounded-md p-4'>
                              <div className='flex items-center gap-3 mb-3'>
                                    <FaExclamationTriangle className='text-
yellow-400 text-xl' />
                                    <h3 className='text-xl font-bold text-
white'>Allergen Information</h3>
                              </div>
                              <p className='text-white text-sm mb-3'>This
product contains the following allergens:</p>
                              <div className='flex flex-wrap gap-2'>
                                    {allergens.map((allergen: string, index:
number) => (
                                          <span
                                                key={index}
                                                className='px-2.5 py-1 bg-
yellow-600 text-white rounded-md text-xs font-medium'>
                                                {allergen}
                                          </span>
                                    ))}
                              </div>
                        </article>
                  )}

                  {/* Nutritional Information Card */}
                  <div className='bg-black/20 rounded-md p-4'>
                        <h3 className='text-xl font-bold text-white mb-4 text-
center'>Nutritional Information</h3>
                        <div className='grid grid-cols-2 sm:grid-cols-3 gap-4'>
                              {Object.entries(nutritionalInfo).map(([key,
value]) => (
                                    <div
                                          key={key}
                                          className='text-center bg-
yellow-500/20 p-3 rounded-md'>
                                          <div className='text-2xl font-bold
text-white'>{value}</div>
                                          <div className='text-sm text-white
capitalize'>{key}</div>
                                    </div>
                              ))}
                        </div>
                  </div>
            </div>
      );
};

export default ProductInfo;
</file>

<file path="client/src/app/menu/[Products]/components/CommentsButton.tsx">
'use client';

import React, { useState, useEffect } from 'react';
import { BiSolidComment } from 'react-icons/bi';
import Button from '@/components/ui/Button';
import { useAuth } from '@/lib/supabase/auth/useAuth';
import { getCommentsForProduct } from '@/lib/supabase/comments';

import CommentModal from './CommentModal';

interface CommentsButtonsProps {
      product: ProductType;
```

```
}

const CommentsButton: React.FC<CommentsButtonsProps> = ({ product }) => {
      const [commentCount, setCommentCount] = useState(0);
      const [showCommentModal, setShowCommentModal] = useState(false);
      const [loading, setLoading] = useState(true);
      const { user } = useAuth();

      // Determine product id/key used in this app (fallbacks for different
shapes)
      const productKey = (product as any).ProductID || (product as any).id ||
(product as any).Name || (product as any).slug || (product as any).Slug;

      const handleCommentClick = () => {
            const userId = (user as any)?.id || (user as any)?.uid || null;
            if (!userId) {
                  alert('Please login to comment');
                  return;
            }
            setShowCommentModal(true);
      };

      const handleCloseCommentModal = () => {
            setShowCommentModal(false);
      };

      useEffect(() => {
            let mounted = true;
            const load = async () => {
                  setLoading(true);
                  try {
                        if (!productKey) {
                              setCommentCount(0);
                              setLoading(false);
                              return;
                        }
                        const comments = await
getCommentsForProduct(productKey);
                        if (!mounted) return;
                        setCommentCount(Array.isArray(comments) ?
comments.length : 0);
                  } catch (err) {
                        console.error('Failed to load comments', err);
                  }
                  setLoading(false);
            };
            load();
            return () => {
                  mounted = false;
            };
      }, [productKey]);

      if (showCommentModal) {
            return (
                  <CommentModal
                        handleCloseCommentModal={handleCloseCommentModal}
                        product={product}
                  />
            );
      }

      return (
            <Button
                  disabled={loading}
```

```
                loading={loading}
                variant='icon'
                onClick={handleCommentClick}
                className='text-white'>
                <BiSolidComment size={20} />
                <span className='text-xs text-center truncate'>
                      {commentCount} Comment{commentCount !== 1 ? 's' : ''}
                </span>
            </Button>
      );
};

export default CommentsButton;
</file>

<file path="client/src/app/photo/booking/page.tsx">
'use client';
import { createClient } from '@/lib/supabase/client';

import Main from '@/components/ui/layout/Main';
import { HiCamera } from 'react-icons/hi2';
import { useForm } from 'react-hook-form';
import { useState } from 'react';

type FormData = {
    name: string;
    email: string;
    phone: string;
    date: string;
    time: string;
    package: string;
    people: number;
    message: string;
};

export default function PhotoBoothBookingPage() {
    const [isSubmitting, setIsSubmitting] = useState(false);
    const [submitStatus, setSubmitStatus] = useState<{ success: boolean;
message: string } | null>(null);

    const { register, handleSubmit, formState: { errors }, reset } =
useForm<FormData>();

    const onSubmit = async (data: FormData) => {
        setIsSubmitting(true);
        try {
            const supabase = createClient();
            const { error } = await
supabase.from('photo_boot_bookings').insert({
                name: data.name,
                email: data.email,
                phone: data.phone,
                date: data.date,
                time: data.time,
                package: data.package,
                people: data.people,
                message: data.message
            } as any);

            if (error) throw error;

            setSubmitStatus({
                success: true,
                message: 'Your booking request has been submitted successfully!
```

```
We\'ll confirm your booking via email within 24 hours.'
            });
            reset();
        } catch (error) {
            console.error('Error submitting booking:', error);
            setSubmitStatus({
                success: false,
                message: 'Failed to submit booking request. Please try again
later.'
            });
        } finally {
            setIsSubmitting(false);
        }
    };

    return (
        <Main
            tittle='Book Your 360 Booth'
            Icon={HiCamera}
            heading='Reserve Your Spot for an Unforgettable Experience'>
            <div className='max-w-2xl mx-auto'>
                <div className='bg-gradient-to-br from-yellow-900/20 to-
amber-900/20 backdrop-blur-md border border-yellow-500/30 rounded-xl p-8 shadow-
xl'>
                    <h2 className='text-2xl font-bold text-white mb-6'>Booking
Form</h2>

                    {submitStatus && (
                        <div className={`mb-6 p-4 rounded-lg ${
{submitStatus.success ? 'bg-green-500/20 border border-green-500/30' : 'bg-
red-500/20 border border-red-500/30'}`}>
                            <p className={submitStatus.success ? 'text-
green-200' : 'text-red-200'}>
                                {submitStatus.message}
                            </p>
                        </div>
                    )}

                    <form onSubmit={handleSubmit(onSubmit)} className='space-
y-6'>
                        {/* Name */}
                        <div>
                            <label className='block text-sm font-medium text-
yellow-300 mb-2' htmlFor='name'>
                                Full Name *
                            </label>
                            <input
                                type='text'
                                id='name'
                                {...register('name', { required: 'Name is
required' })}
                                className='w-full rounded-lg border border-
white/30 bg-yellow-900/60 text-white px-4 py-2.5 outline-none transition
focus:border-amber-400 focus:ring-2 focus:ring-amber-400/40'
                                placeholder='John Doe'
                            />
                            {errors.name && <p className='mt-1 text-sm text-
red-300'>{errors.name.message}</p>}
                        </div>

                        {/* Email */}
                        <div>
                            <label className='block text-sm font-medium text-
yellow-300 mb-2' htmlFor='email'>
```

```
                                Email Address *
                            </label>
                            <input
                                type='email'
                                id='email'
                                {...register('email', {
                                    required: 'Email is required',
                                    pattern: {
                                        value: /^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-
Z]{2,}$/i,
                                        message: 'Invalid email address'
                                    }
                                })}
                                className='w-full rounded-lg border border-
white/30 bg-yellow-900/60 text-white px-4 py-2.5 outline-none transition
focus:border-amber-400 focus:ring-2 focus:ring-amber-400/40'
                                placeholder='john@example.com'
                            />
                            {errors.email && <p className='mt-1 text-sm text-
red-300'>{errors.email.message}</p>}
                        </div>

                        {/* Phone */}
                        <div>
                            <label className='block text-sm font-medium text-
yellow-300 mb-2' htmlFor='phone'>
                                Phone Number *
                            </label>
                            <input
                                type='tel'
                                id='phone'
                                {...register('phone', { required: 'Phone number
is required' })}
                                className='w-full rounded-lg border border-
white/30 bg-yellow-900/60 text-white px-4 py-2.5 outline-none transition
focus:border-amber-400 focus:ring-2 focus:ring-amber-400/40'
                                placeholder='+27 12 345 6789'
                            />
                            {errors.phone && <p className='mt-1 text-sm text-
red-300'>{errors.phone.message}</p>}
                        </div>

                        {/* Date */}
                        <div>
                            <label className='block text-sm font-medium text-
yellow-300 mb-2' htmlFor='date'>
                                Preferred Date *
                            </label>
                            <input
                                type='date'
                                id='date'
                                {...register('date', { required: 'Date is
required' })}
                                className='w-full rounded-lg border border-
white/30 bg-yellow-900/60 text-white px-4 py-2.5 outline-none transition
focus:border-amber-400 focus:ring-2 focus:ring-amber-400/40'
                            />
                            {errors.date && <p className='mt-1 text-sm text-
red-300'>{errors.date.message}</p>}
                        </div>

                        {/* Time */}
                        <div>
                            <label className='block text-sm font-medium text-
```

```
yellow-300 mb-2' htmlFor='time'>
                                Preferred Time *
                            </label>
                            <input
                                type='time'
                                id='time'
                                {...register('time', { required: 'Time is
required' })}
                                className='w-full rounded-lg border border-
white/30 bg-yellow-900/60 text-white px-4 py-2.5 outline-none transition
focus:border-amber-400 focus:ring-2 focus:ring-amber-400/40'
                            />
                            {errors.time && <p className='mt-1 text-sm text-
red-300'>{errors.time.message}</p>}
                        </div>

                        {/* Package Type */}
                        <div>
                            <label className='block text-sm font-medium text-
yellow-300 mb-2' htmlFor='package'>
                                Package Type *
                            </label>
                            <select
                                id='package'
                                {...register('package', { required: 'Please
select a package' })}
                                className='w-full rounded-lg border border-
white/30 bg-yellow-900/60 text-white px-4 py-2.5 outline-none transition
focus:border-amber-400 focus:ring-2 focus:ring-amber-400/40'>
                                <option value=''>Select a package</option>
                                <option value='individual'>Individual Session
(R50/person)</option>
                                <option value='event'>Event Package (Contact for
pricing)</option>
                            </select>
                            {errors.package && <p className='mt-1 text-sm text-
red-300'>{errors.package.message}</p>}
                        </div>

                        {/* Number of People */}
                        <div>
                            <label className='block text-sm font-medium text-
yellow-300 mb-2' htmlFor='people'>
                                Number of People
                            </label>
                            <input
                                type='number'
                                id='people'
                                {...register('people', {
                                    min: { value: 1, message: 'Minimum 1 person
required' },
                                    valueAsNumber: true
                                })}
                                min='1'
                                defaultValue='1'
                                className='w-full rounded-lg border border-
white/30 bg-yellow-900/60 text-white px-4 py-2.5 outline-none transition
focus:border-amber-400 focus:ring-2 focus:ring-amber-400/40'
                            />
                            {errors.people && <p className='mt-1 text-sm text-
red-300'>{errors.people.message}</p>}
                        </div>

                        {/* Special Requests */}
```

```
                        <div>
                            <label className='block text-sm font-medium text-
yellow-300 mb-2' htmlFor='message'>
                                Special Requests or Notes
                            </label>
                            <textarea
                                id='message'
                                {...register('message')}
                                rows={4}
                                className='w-full rounded-lg border border-
white/30 bg-yellow-900/60 text-white px-4 py-2.5 outline-none transition
focus:border-amber-400 focus:ring-2 focus:ring-amber-400/40'
                                placeholder='Any special requirements or
questions?'
                            />
                        </div>

                        {/* Submit Button */}
                        <button
                            type='submit'
                            disabled={isSubmitting}
                            className='w-full px-6 py-3 bg-gradient-to-r from-
amber-600 to-yellow-600 hover:from-amber-500 hover:to-yellow-500 rounded-xl
shadow-lg hover:shadow-amber-500/50 transition-all duration-200 font-semibold
text-white disabled:opacity-70 disabled:cursor-not-allowed'>
                            {isSubmitting ? 'Submitting...' : 'Submit Booking
Request'}
                        </button>

                        <p className='text-sm text-yellow-400 text-center'>
                            * Required fields. We'll confirm your booking via
email within 24 hours.
                        </p>
                    </form>
                </div>
            </div>
        </Main>
    );
}
</file>

<file path="client/src/app/privacy/components/PrivacyContent.tsx">
import Section from '@/components/ui/layout/Section';

const PrivacyContent: React.FC = () => {
      return (
            <Section tittle='Privacy policy '>
                  <article>
                        <div className='space-y-4'>
                              <p>Privacy policy details will be listed here.</p>
                        </div>
                  </article>
            </Section>
      );
};

export default PrivacyContent;
</file>

<file path="client/src/app/terms/components/TermsContent.tsx">
import Section from '@/components/ui/layout/Section';

const TermsContent: React.FC = () => {
      return (
```

```
            <Section>
                  <div className='bg-yellow-700 p-6 sm:p-8 rounded-md text-
white'>
                        <header className='text-center mb-8'>
                              <h1
                                    className='text-3xl sm:text-4xl font-bold'
                                    style={{ filter: 'drop-shadow(0 0 4px
#f0b100)' }}>
                                    Terms and Conditions
                              </h1>
                              <p className='text-white mt-2'>Last updated: July
24, 2024</p>
                        </header>

                        <div className='space-y-6'>
                              <div className='bg-black/20 p-4 rounded-md'>
                                    <h2 className='text-xl font-semibold
mb-3'>1. Introduction</h2>
                                    <p className='text-white'>
                                          Welcome to PA Luxe Creation. These
terms and conditions outline the rules and
                                          regulations for the use of our
services. By accessing this website and using our
                                          services, you accept these terms and
conditions in full.
                                    </p>
                              </div>

                              <div className='bg-black/20 p-4 rounded-md'>
                                    <h2 className='text-xl font-semibold
mb-3'>2. User Responsibilities</h2>
                                    <p className='text-white mb-2'>
                                          You are responsible for ensuring that
your account information is accurate and for
                                          maintaining the confidentiality of
your password. You agree to accept responsibility
                                          for all activities that occur under
your account.
                                    </p>
                                    <ul className='list-disc list-inside space-
y-1 text-white pl-2'>
                                          <li>You must not misuse our
services.</li>
                                          <li>You must not attempt to gain
unauthorized access to our systems.</li>
                                          <li>You must comply with all
applicable laws and regulations.</li>
                                    </ul>
                              </div>

                              <div className='bg-black/20 p-4 rounded-md'>
                                    <h2 className='text-xl font-semibold
mb-3'>3. Limitation of Liability</h2>
                                    <p className='text-white'>
                                          In no event shall PA Luxe Creation,
nor any of its officers, directors, and employees,
                                          be liable to you for anything arising
out of or in any way connected with your use of
                                          this website, whether such liability
is under contract, tort or otherwise.
                                    </p>
                              </div>

                              <div className='bg-black/20 p-4 rounded-md'>
```

```
                                    <h2 className='text-xl font-semibold
mb-3'>4. Governing Law</h2>
                                    <p className='text-white'>
                                          These Terms will be governed by and
construed in accordance with the laws of South
                                          Africa, and you submit to the non-
exclusive jurisdiction of the state and federal
                                          courts located in Limpopo for the
resolution of any disputes.
                                    </p>
                              </div>
                        </div>
                  </div>
            </Section>
      );
};

export default TermsContent;
</file>

<file path="client/src/app/terms/page.tsx">
import { Suspense } from 'react';
import Loading from '@/components/ui/Loading';
import Main from '@/components/ui/layout/Main';
import { FaUserCircle } from 'react-icons/fa';
import Image from 'next/image';
import Link from 'next/link';
import AppLink from '@/components/ui/Link';
import Section from '@/components/ui/layout/Section';
import { BiX } from 'react-icons/bi';

export default function TermsAndConditionsPage() {
      const team = [
            {
                  name: 'Ntsako Nyath',
                  role: 'Maneger (Industrial Engineering)',
                  image: '/users/Abram.jpg',
                  bio: 'My name is Abram Elton Ntsako Nyathi and I live in
Phalaborwa Evander. Iâ ™m passionate about chess and enjoy playing it in my free
time. Iâ ™m also interested in entrepreneurship and my goal is to have and run
my own company. I have experience in coding (HTML, CSS, JavaScript), CAD (Fusion
360), and editing (Photoshop, Illustrator, After Effects). Iâ ™m proficient in
Microsoft Word, Excel, and PowerPoint. Iâ ™m currently a student at Vaal
University of Technology studying Industrial Engineering. Here are some of the
subjects Iâ ™ve taken so far:',
                  link: {
                        Link: 'http://abrameltonntsako.web.app',
                        Name: 'abrameltonntsako.web.app',
                  },
            },
      ];
      return (
            <Main
                  tittle='Terms and Conditions'
                  Icon={FaUserCircle}
                  heading='Our Centarl Eatery Terms and Conditions'>
                  <Section className='grid md:grid-cols-2 gap-12'>
                        {team.map((member, index) => (
                              <div
                                    key={member.name}
                                    className='group'>
                                    <div className='relative w-[300]
rounded-2xl overflow-hidden p-0 mb-8 border border-[var(--glass-border)] shadow-
xl'>
```

```
                                        <Image
                                              src={member.image}
                                              alt={member.name}
                                              width={300}
                                              height={300}
                                              className=' rounded-md
transition-all duration-500 group-hover:scale-110'
                                        />
                                        <div className='absolute inset-0 bg-
gradient-to-t from-black/60 via-black/20 to-transparent opacity-0 group-
hover:opacity-100 transition-opacity duration-300' />
                                  </div>
                                  <div className='text-center'>
                                        <h3 className='text-2xl font-bold mb-3
text-high-contrast group-hover:text-[var(--color-yellow-light)] transition-
colors duration-300 drop-shadow-text'>
                                              {member.name}
                                        </h3>
                                        <p className='text-lg font-medium
text-[var(--color-yellow)] mb-4 drop-shadow'>
                                              {member.role}
                                        </p>
                                        <p className='text-high-contrast
leading-relaxed'>{member.bio}</p>
                                        <AppLink
                                              variant='primary'
                                              href={member.link.Link}>
                                              {member.link.Name}
                                        </AppLink>
                                  </div>
                                  <div className='mt-6 flex justify-center
space-x-4'>
                                        <a
                                              href='#'
                                              className='text-high-contrast/70
hover:text-[var(--color-yellow-light)] transition-colors duration-300
hover:drop-shadow-glow'>
                                              <BiX />
                                              <svg
                                                    className='w-6 h-6'
                                                    fill='currentColor'
                                                    viewBox='0 0 24 24'>
                                                    <path d='M19 0h-14c-2.761
0-5 2.239-5 5v14c0 2.761 2.239 5 5 5h14c2.762 0 5-2.239 5-5v-14c0-2.761-2.238-5-
5-5zm-11 19h-3v-11h3v11zm-1.5-12.268c-.966 0-1.75-.79-1.75-1.764s.784-1.764
1.75-1.764 1.75.79 1.75 1.764-.783 1.764-1.75 1.764zm13.5 12.268h-3v-5.604c0-
3.368-4-3.113-4 0v5.604h-3v-11h3v1.765c1.396-2.586 7-2.777 7 2.476v6.759z' />
                                              </svg>
                                        </a>
                                        <a
                                              href='#'
                                              className='text-high-contrast/70
hover:text-[var(--color-yellow-light)] transition-colors duration-300
hover:drop-shadow-glow'>
                                              <svg
                                                    className='w-6 h-6'
                                                    fill='currentColor'
                                                    viewBox='0 0 24 24'>
                                                    <path d='M23 3a10.9 10.9 0
0 1-3.14 1.53 4.48 4.48 0 0 0-7.86 3v1A10.66 10.66 0 0 1 3 4s-4 9 5 13a11.64
11.64 0 0 1-7 2c9 5 20 0 20-11.5a4.5 4.5 0 0 0-.08-.83A7.72 7.72 0 0 0 23 3z' />
                                              </svg>
                                        </a>
                                  </div>
```

```
                            </div>
                        ))}
                    </Section>
                </Main>
        );
}
</file>

<file path="client/src/components/components.css">
.spinner {
  width: 48px;
  height: 48px;
  border: 6px solid rgba(255, 255, 255, 0.3);
  border-top-color: #f0b100;
  border-radius: 50%;
  animation: spin 1s linear infinite;
  margin-bottom: 16px;
}

@keyframes spin {
  to { transform: rotate(360deg); }
}
</file>

<file path="client/src/components/ui/Alert.tsx">
import React from 'react';
import { FaCheckCircle, FaExclamationTriangle, FaTimes } from 'react-icons/fa';

interface AlertProps {
  variant: 'success' | 'danger';
  onClose?: () => void;
  className?: string;
  children?: React.ReactNode;
  message?: string; // Keeping for backward compatibility
}

const Alert: React.FC<AlertProps> = ({
  variant,
  onClose,
  className = '',
  children,
  message,
}) => {
  const styles = {
    success: 'bg-green-500/20 text-white',
    danger: 'bg-yellow-500/20 text-white',
  };

  const displayMessage = children || message;

  return (
    <div
      className={`mt-4 p-3 rounded-md flex items-center gap-2 relative ${
        styles[variant]
      } ${className}`}
    >
      {variant === 'success' ? <FaCheckCircle /> : <FaExclamationTriangle />}
      <span className='flex-1'>{displayMessage}</span>
      {onClose && (
        <button
          onClick={onClose}
          className='text-white hover:opacity-75 transition-opacity'
          aria-label='Close alert'
        >
```

```
          <FaTimes />
        </button>
      )}
    </div>
  );
};

export default Alert;
</file>

<file path="client/src/components/ui/Avatar.tsx">
import React from 'react';
import classNames from 'classnames'; // Import classNames
import Image from 'next/image';

interface AvatarProps {
      src: string | null | undefined;
      alt: string;
      size?: 'small' | 'medium' | 'large';
      className?: string; // Existing className for external styles
}

// Define size classes as an object for better readability and maintainability
const sizeClassesMap = {
      small: 'w-8 h-8 text-sm', // Added text-sm for initial/fallback text size
      medium: 'w-10 h-10 text-base', // Added text-base
      large: 'w-16 h-16 text-xl', // Added text-xl
};

const Avatar: React.FC<AvatarProps> = ({ src, alt, size = 'medium', className })
=> {
      // Get the size-specific classes from the map, defaulting to 'medium'
      const currentSizeClasses = sizeClassesMap[size];

      return (
            <div
                  className={classNames(
                        'relative flex items-center justify-center rounded-full
bg-yellow-200 overflow-hidden',
                        currentSizeClasses, // Apply the size-specific classes
                        className // Apply any additional classes passed from
parent
                  )}>
                  {src ? (
                        <Image
                              src={src}
                              alt={alt}
                              className='absolute inset-0 w-full h-full object-
cover'
                        />
                  ) : (
                        // Fallback for no image: display first letter of alt
text
                        <span className='text-yellow-600 font-bold uppercase
flex items-center justify-center h-full w-full'>
                              {alt.charAt(0)}
                        </span>
                  )}
            </div>
      );
};

export default Avatar;
</file>
```

```
<file path="client/src/components/ui/Icon.tsx">
import React from 'react';

interface IconProps {
      icon: React.ElementType;
      className?: string;
      variant?: 'default' | 'inline' | 'inlineCircular';
      heading?: string;
}

const Icon: React.FC<IconProps> = ({
      icon: IconComponent,
      className,
      variant = 'default',
      heading,
}) => {
      if (variant === 'inline') {
            return (
                  <div className='flex items-center justify-center gap-2'>
                        <IconComponent className={`${className} text-3xl text-
yellow-500`} />
                        {heading && <h2 className='border-none'>{heading}</h2>}
                  </div>
            );
      }

      if (variant === 'inlineCircular') {
            return (
                  <div className='flex items-center  gap-2 '>
                        <span className='bg-yellow-500 shadow-sm shadow-black/50
rounded-full p-2 '>
                              <IconComponent className={` text-white text-xl   $
{className || ''}`} />
                        </span>

                        {heading && <h3 className='text-shadow-sm text-shadow-
black/50 p-2'>{heading}</h3>}
                  </div>
            );
      }

      // Default variant (circular icon)
      return (
            <div className='flex flex-col items-center text-center'>
                  <div className='flex items-center justify-center w-12 h-12 bg-
yellow-500 mx-auto text-white rounded-full shadow-sm shadow-black/50 '>
                        <IconComponent className={`w-6 h-6 ${className || ''}
`} />
                  </div>
                  {heading && <h3 className='mt-4 text-shadow-sm text-shadow-
black/50'>{heading}</h3>}
            </div>
      );
};

export default Icon;
</file>

<file path="client/src/components/ui/TimeSlotPicker.tsx">
import React from 'react';

const timeSlots = ['09:00', '10:00', '11:00', '12:00', '13:00', '14:00',
'15:00', '16:00'];
```

```
interface TimeSlotPickerProps {
      selectedTime: string;
      onChange: (time: string) => void;
      availableWorkers: string[];
      disabled?: boolean;
}

const TimeSlotPicker: React.FC<TimeSlotPickerProps> = ({
      selectedTime,
      onChange,
      availableWorkers,
      disabled = false,
}) => (
      <div className='space-y-4'>
            <h3 className='text-lg font-semibold text-white'>Available Time
Slots</h3>
            <div className='grid grid-cols-2 sm:grid-cols-4 gap-3'>
                  {timeSlots.map((time) => {
                        const isAvailable = !disabled && availableWorkers.length
> 0;
                        return (
                              <button
                                    key={time}
                                    type='button'
                                    onClick={() => isAvailable &&
onChange(time)}
                                    disabled={!isAvailable}
                                    className={`p-3 rounded-md text-center
transition-colors ${
                                          selectedTime === time
                                                ? 'bg-yellow-500 text-white'
                                                : isAvailable
                                                ? 'bg-black/30 text-white
hover:bg-yellow-500/20'
                                                : 'bg-black/10 text-white/40
cursor-not-allowed'
                                    }`}>
                                    {time}
                              </button>
                        );
                  })}
            </div>
            {!disabled && availableWorkers.length === 0 && (
                  <p className='text-yellow-400'>No workers available for these
time slots.</p>
            )}
      </div>
);

export default TimeSlotPicker;
</file>

<file path="client/src/lib/context/CartContext.tsx">
'use client';

import { createContext, useContext, useState, ReactNode, SetStateAction,
Dispatch } from 'react';



interface CartContextType {
      cartItems: CartItem[];
      addItem: (item: ProductType) => void;
```

```
      removeFromCart: (ProductID: ProductType['ProductID']) => void;
      clearCart: () => void;
      minisItemQuantity: (item: ProductType) => void;
      totalPrice: number;
      totalQuantity: number;
}

export const useCart = () => {
      const context = useContext(CartContext);
      if (context === undefined) {
            throw new Error('useCart must be used within a CartProvider');
      }
      return context;
};

const CartContext = createContext<CartContextType | undefined>(undefined);

export const CartProvider = ({ children }: { children: ReactNode }) => {
      const [cartItems, setCartItems] = useState<CartItem[]>([]);

      const addItem = (item: ProductType) => {
            setCartItems((prevItems) => {
                  const existingItem = prevItems.find((cartItem) =>
cartItem.ProductID === item.ProductID);
                  if (existingItem) {
                        return prevItems.map((cartItem) =>
                              cartItem.ProductID === item.ProductID
                                    ? { ...cartItem, quantity: cartItem.quantity
+ 1 }
                                    : cartItem
                        );
                  } else {
                        const newItem: CartItem = { ...item, quantity: 1 };
                        return [...prevItems, newItem];
                  }
            });
      };

      const removeFromCart = (ProductID: ProductType['ProductID']) => {
            const updatedItems = cartItems.filter((item) => item.ProductID !==
ProductID);
            setCartItems(updatedItems);
      };

      const minisItemQuantity = (item: ProductType) => {
            setCartItems((prevItems) => {
                  return prevItems.map((cartItem) =>
                        cartItem.Name === item.Name ? { ...cartItem, quantity:
cartItem.quantity - 1 } : cartItem
                  );
            });
      };

      const clearCart = () => {
            setCartItems([]);
      };

      const totalPrice = cartItems.reduce((sum, item) => sum + item.Price *
item.quantity, 0);
      const totalQuantity = cartItems.reduce((sum, item) => sum + item.quantity,
0);

      return (
            <CartContext.Provider
```

```
                  value={{
                        cartItems,
                        totalPrice,
                        totalQuantity,
                        addItem,
                        removeFromCart,
                        minisItemQuantity,
                        clearCart,
                  }}>
                  {children}
            </CartContext.Provider>
      );
};

// Ensure CartItem has ProductID, Name, Price, quantity, and Image properties

// Toast Context for notifications
interface ToastContextType {
      showToast: (message: string) => void;
}
const ToastContext = createContext<ToastContextType | undefined>(undefined);

export const ToastProvider = ({ children }: { children: ReactNode }) => {
      const [toast, setToast] = useState<string | null>(null);
      const showToast = (message: string) => {
            setToast(message);
            setTimeout(() => setToast(null), 2500); // 4 seconds
      };
      return (
            <ToastContext.Provider value={{ showToast }}>
                  {children}
                  {toast && (
                        <div className='fixed bottom-8 right-8 z-50 bg-black
border-2 border-yellow-500  font-bold text-lg px-6 py-3 rounded-md shadow-2xl
animate-fadein min-w-[180px] text-center'>
                              {toast}
                        </div>
                  )}
            </ToastContext.Provider>
      );
};

export const useToast = () => {
      const context = useContext(ToastContext);
      if (!context) throw new Error('useToast must be used within a
ToastProvider');
      return context;
};
</file>

<file path="client/src/lib/supabase/orders.ts">
import { createClient } from './client';

const supabaseBrowser = createClient();

export const addOrder = async (userId: string | null, items: any[], totalPrice:
number, totalQuantity: number) => {
  const payload = {
    user_id: userId,
    items,
    total_price: totalPrice,
    total_quantity: totalQuantity,
  };
  const { data, error } = await supabaseBrowser
```

```
    .from('orders')
    // @ts-ignore - Supabase type inference issue with Database types
    .insert([payload])
    .select()
    .single();
  if (error) throw error;
  return data;
};

export const getOrdersByUser = async (userId: string) => {
  const { data, error } = await supabaseBrowser
    .from('orders')
    .select('*')
    .eq('user_id', userId)
    .order('created_at', { ascending: false });
  if (error) throw error;
  return data;
};
</file>

<file path="client/src/lib/types/ProductTypes.d.ts">
declare global {
  interface nutritionalInfo {
    calories: number;
    protein: string;
    carbs: string;
    fat: string;
    fiber: string;
    sugar: string;
  }

  interface ProductType {
    ProductID: string;
    Name: string;
    Description: string[];
    Price: number;
    Image?: string;
    badge?: string;
    rating?: number;
    nutritional?: nutritionalInfo;
    allergens?: string[];
    isVegetarian?: boolean;
    isSpicy?: boolean;
  }

  type ProductsType = {
    id: string | number;
    Name: string;
    Products: ProductType[];
    Image?: string;
    Description?: string;
  }[];
}
export { };
</file>

<file path="client/src/proxy.ts">
import { createServerClient } from '@supabase/ssr';
import { NextResponse, type NextRequest } from 'next/server';

export async function proxy(request: NextRequest) {
    let response = NextResponse.next({
        request: {
            headers: request.headers,
```

```
        },
    });

    const supabase = createServerClient(
        process.env.NEXT_PUBLIC_SUPABASE_URL!,
        process.env.NEXT_PUBLIC_SUPABASE_PUBLISHABLE_KEY!,
        {
            cookies: {
                getAll() {
                    return request.cookies.getAll();
                },
                setAll(cookiesToSet) {
                    cookiesToSet.forEach(({ name, value, options }) => {
                        request.cookies.set(name, value);
                        response.cookies.set(name, value, options);
                    });
                },
            },
        }
    );

    const {
        data: { user },
    } = await supabase.auth.getUser();
    const authPaths = ['/login', '/register']
    const protectedPaths = ['/profile', '/orders', '/photo/booking'];
    const isProtected = protectedPaths.some((path) =>
        request.nextUrl.pathname.startsWith(path)
    );
    const isAuthPage = authPaths.some((path) =>
        request.nextUrl.pathname.startsWith(path)
    );

    if (isProtected && !user) {
        const url = request.nextUrl.clone();
        url.pathname = '/login';
        url.searchParams.set('redirectTo', request.nextUrl.pathname);

        // Create a redirect response
        const redirectResponse = NextResponse.redirect(url);

        // Copy cookies from the original response (which might have refreshed
tokens)
        // to the redirect response
        response.cookies.getAll().forEach((cookie) => {
            redirectResponse.cookies.set(cookie.name, cookie.value, cookie);
        });

        return redirectResponse;
    }
    if (isAuthPage && user) {
        const url = request.nextUrl.clone();
        url.pathname = '/profile';

        // Create a redirect response
        const redirectResponse = NextResponse.redirect(url);

        // Copy cookies from the original response (which might have refreshed
tokens)
        // to the redirect response
        response.cookies.getAll().forEach((cookie) => {
            redirectResponse.cookies.set(cookie.name, cookie.value, cookie);
        });
```

```
        return redirectResponse;
    }

    return response;
}

export const config = {
    matcher: [
        /*
         * Match all request paths except for the ones starting with:
         * - _next/static (static files)
         * - _next/image (image optimization files)
         * - favicon.ico (favicon file)
         * Feel free to modify this pattern to include more paths.
         */
        '/((?!_next/static|_next/image|favicon.ico|.*\\.(?:svg|png|jpg|jpeg|gif|
webp)$).*)',
    ],
};
</file>

<file path="admin/src/components/layout/AdminSidebar.tsx">
"use client";

import Link from "next/link";
import { useState } from "react";
import { HiOutlineMenu, HiOutlineX } from "react-icons/hi";
import { NAV_ITEMS } from "./nav-items";
import { SidebarNav } from "./SidebarNav";

export const AdminSidebar = () => {
  const [mobileMenuOpen, setMobileMenuOpen] = useState(false);

  return (
    <>
      {/* Mobile header with burger menu */}
      <header className="fixed left-0 right-0 top-0 z-50 border-b
border-white/10 bg-yellow-950/95 backdrop-blur-sm lg:hidden">
        <div className="flex items-center justify-between px-4 py-3">
          <button
            type="button"
            onClick={() => setMobileMenuOpen(!mobileMenuOpen)}
            className="rounded-lg bg-yellow-900/90 p-2 text-white hover:bg-
yellow-800"
            aria-label="Toggle menu"
          >
            {mobileMenuOpen ? (
              <HiOutlineX className="h-6 w-6" />
            ) : (
              <HiOutlineMenu className="h-6 w-6" />
            )}
          </button>

          <div className="text-center">
            <div className="text-xs uppercase tracking-[0.3em] text-indigo-400">
              PA Catering
            </div>
            <p className="text-sm font-semibold text-white">Admin</p>
          </div>

          <div className="w-10" /> {/* Spacer for centering */}
        </div>
      </header>
```

```
      {/* Mobile overlay */}
      {mobileMenuOpen && (
        <div
          className="fixed inset-0 z-30 bg-black/60 backdrop-blur-sm lg:hidden"
          onClick={() => setMobileMenuOpen(false)}
        />
      )}

      {/* Sidebar */}
      <aside
        className={`fixed inset-y-0 left-0 z-40 w-72 flex-col border-r border-
white/10 bg-yellow-950/95 px-6 py-8 backdrop-blur-sm transition-transform
duration-300 lg:static lg:flex ${mobileMenuOpen ? "flex translate-x-0" : "-
translate-x-full lg:translate-x-0"
          }`}
      >
        <Link
          href="/dashboard"
          className="mb-10 block"
          onClick={() => setMobileMenuOpen(false)}
        >
          <div className="text-xs uppercase tracking-[0.4em] text-indigo-400">
            PA Catering
          </div>
          <p className="text-xl font-semibold text-white">Admin Console</p>
        </Link>

        <div onClick={() => setMobileMenuOpen(false)}>
          <SidebarNav items={NAV_ITEMS} />
        </div>

        <div className="mt-auto rounded-xl border border-white/10 bg-
yellow-900/80 p-4 text-sm text-yellow-300">
          <p className="font-semibold text-white">Daily Ops Checklist</p>
          <ul className="mt-2 space-y-1 text-xs text-yellow-400">
            <li>â ¢ Confirm menu availability</li>
            <li>â ¢ Assign pending orders</li>
            <li>â ¢ Update stock counts</li>
          </ul>
        </div>
      </aside>
    </>
  );
};
</file>

<file path="admin/src/components/menu/products/ProductEditForm.tsx">
"use client";

import { useRouter } from "next/navigation";
import { useFormState } from "react-dom";
import { SubmitButton } from "@/components/forms/SubmitButton";
import {
    type ProductActionState,
    updateProductAction,
    deleteProductAction,
} from "@/lib/data/products-actions";
import type { ProductRecord } from "@/lib/types";
import { useState } from "react";
import { DeleteConfirmDialog } from "@/components/shared/DeleteConfirmDialog";
import { ImageUpload } from "@/components/forms/ImageUpload";
import { deleteImage } from "@/lib/supabase/storage";

type Props = {
```

```
    product: ProductRecord;
    categories: string[];
};

const initialState: ProductActionState = {};

export const ProductEditForm = ({ product, categories }: Props) => {
    const router = useRouter();
    const [state, formAction] = useFormState(updateProductAction, initialState);
    const [showDeleteDialog, setShowDeleteDialog] = useState(false);
    const [uploadedPath, setUploadedPath] = useState<string>("");

    const handleDelete = async () => {
        await deleteProductAction(product.id);
        router.push("/menu");
    };

    const handleCancel = async () => {
        if (uploadedPath) {
            try {
                await deleteImage(uploadedPath);
            } catch (error) {
                console.error("Failed to delete uploaded image:", error);
            }
        }
        router.back();
    };

    return (
        <div className="space-y-6">
            <form action={formAction} className="rounded-2xl border border-
white/10 bg-yellow-900/40 p-6 space-y-4">
                <input type="hidden" name="id" value={product.id} />

                <div className="grid gap-4 md:grid-cols-2">
                    <label className="space-y-2 text-sm">
                        <span className="text-yellow-300">Product Name *</span>
                        <input
                            name="name"
                            type="text"
                            required
                            defaultValue={product.name}
                            className="w-full rounded-lg border border-white/10
bg-yellow-900/60 px-3 py-2 text-white outline-none focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
                        />
                    </label>

                    <label className="space-y-2 text-sm">
                        <span className="text-yellow-300">Category *</span>
                        <select
                            name="category"
                            required
                            defaultValue={product.category_name ?? ""}
                            className="w-full rounded-lg border border-white/10
bg-yellow-900/60 px-3 py-2 text-white outline-none focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
                        >
                            {categories.map((cat) => (
                                <option key={cat} value={cat}>
                                    {cat}
                                </option>
                            ))}
                        </select>
```

```
                    </label>
                </div>

                <div className="grid gap-4 md:grid-cols-2">
                    <label className="space-y-2 text-sm">
                        <span className="text-yellow-300">Price (ZAR) *</span>
                        <input
                            name="price"
                            type="number"
                            step="0.01"
                            min="0"
                            required
                            defaultValue={product.price ?? 0}
                            className="w-full rounded-lg border border-white/10
bg-yellow-900/60 px-3 py-2 text-white outline-none focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
                        />
                    </label>

                    <label className="space-y-2 text-sm">
                        <span className="text-yellow-300">Stock *</span>
                        <input
                            name="stock"
                            type="number"
                            min="0"
                            required
                            defaultValue={product.stock ?? 0}
                            className="w-full rounded-lg border border-white/10
bg-yellow-900/60 px-3 py-2 text-white outline-none focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
                        />
                    </label>
                </div>

                <label className="block space-y-2 text-sm">
                    <span className="text-yellow-300">Product Image</span>
                    <ImageUpload
                        defaultValue={product.image_url}
                        onChange={(url, path) => setUploadedPath(path || "")}
                        folder="products"
                    />
                </label>

                <label className="block space-y-2 text-sm">
                    <span className="text-yellow-300">Badge</span>
                    <input
                        name="badge"
                        defaultValue={product.badge ?? ""}
                        placeholder="Chef's pick, New, Bestseller, etc."
                        className="w-full rounded-lg border border-white/10 bg-
yellow-900/60 px-3 py-2 text-white outline-none focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
                    />
                </label>

                <label className="block space-y-2 text-sm">
                    <span className="text-yellow-300">Description</span>
                    <textarea
                        name="description"
                        rows={4}
                        defaultValue={product.description ?? ""}
                        className="w-full rounded-lg border border-white/10 bg-
yellow-900/60 px-3 py-2 text-white outline-none focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
```

```
                    />
                </label>

                <label className="flex items-center gap-2 text-sm">
                    <input
                        name="is_hidden"
                        type="checkbox"
                        defaultChecked={product.is_hidden ?? false}
                        className="h-4 w-4 rounded border-white/10 bg-
yellow-900/60 text-indigo-600 focus:ring-2 focus:ring-indigo-400/40"
                    />
                    <span className="text-yellow-300">Hide this product from
public view</span>
                </label>

                {state.error && <p className="text-sm text-
rose-400">{state.error}</p>}
                {state.success && (
                    <p className="text-sm text-emerald-400">{state.success}</p>
                )}

                <div className="flex gap-3">
                    <SubmitButton
                        label="Save Changes"
                        loadingLabel="Saving..."
                        className="flex-1 bg-indigo-600 px-4 py-2 text-sm"
                    />
                    <button
                        type="button"
                        onClick={handleCancel}
                        className="rounded-lg bg-yellow-800 px-4 py-2 text-sm
text-white hover:bg-yellow-700"
                    >
                        Cancel
                    </button>
                </div>
            </form>

            {/* Delete Section */}
            <div className="rounded-2xl border border-rose-500/30 bg-rose-500/5
p-6">
                <h3 className="text-lg font-semibold text-white">Danger
Zone</h3>
                <p className="mt-1 text-sm text-yellow-400">
                    Permanently delete this product. This action cannot be
undone.
                </p>
                <button
                    type="button"
                    onClick={() => setShowDeleteDialog(true)}
                    className="mt-4 rounded-lg bg-rose-600 px-4 py-2 text-sm
font-medium text-white hover:bg-rose-700"
                >
                    Delete Product
                </button>
            </div>

            <DeleteConfirmDialog
                isOpen={showDeleteDialog}
                onClose={() => setShowDeleteDialog(false)}
                onConfirm={handleDelete}
                title="Delete Product"
                message={`Are you sure you want to delete "${product.name}"?
This action cannot be undone.`}
```

```
            />
        </div>
    );
};
</file>

<file path="admin/src/components/profiles/ProfilesBoard.tsx">
"use client";

import { useState, useTransition } from "react";
import { useFormState } from "react-dom";
import { HiOutlineEye } from "react-icons/hi2";
import { SubmitButton } from "@/components/forms/SubmitButton";
import { ProfileDetailModal } from "./ProfileDetailModal";
import {
    type ProfileActionState,
    updateProfileAction,
} from "@/lib/data/profiles-actions";
import type { ProfileRecord } from "@/lib/types";
import { formatDistanceToNow } from "date-fns";

type Props = {
    profiles: ProfileRecord[];
    onViewProfile: (profileId: string) => Promise<{
        profile: ProfileRecord;
        orders: any[];
        favorites: any[];
    }>;
};

const initialState: ProfileActionState = {};

export const ProfilesBoard = ({ profiles, onViewProfile }: Props) => {
    const [search, setSearch] = useState("");
    const [editingId, setEditingId] = useState<string | null>(null);
    const [viewingProfile, setViewingProfile] = useState<{
        profile: ProfileRecord;
        orders: any[];
        favorites: any[];
    } | null>(null);
    const [isPending, startTransition] = useTransition();

    const filtered = profiles.filter(
        (profile) =>
            !search ||
            profile.display_name?.toLowerCase().includes(search.toLowerCase())
||
            profile.email?.toLowerCase().includes(search.toLowerCase()) ||
            profile.uid?.toLowerCase().includes(search.toLowerCase()),
    );

    const handleViewProfile = (profileId: string) => {
        startTransition(async () => {
            const data = await onViewProfile(profileId);
            setViewingProfile(data);
        });
    };

    return (
        <div className="space-y-4">
            <div className="flex items-center justify-between gap-3">
                <input
                    type="search"
                    placeholder="Search profiles..."
```

```
                    value={search}
                    onChange={(e) => setSearch(e.target.value)}
                    className="w-full max-w-md rounded-full border border-
white/10 bg-yellow-900/60 px-4 py-3 text-white outline-none focus:border-
indigo-400 focus:ring-2 focus:ring-indigo-400/40"
                />
                <p className="text-xs uppercase tracking-[0.3em] text-
yellow-500">
                    {filtered.length} profiles
                </p>
            </div>

            <div className="grid gap-4 md:grid-cols-2">
                {filtered.map((profile) => (
                    <ProfileCard
                        key={profile.id}
                        profile={profile}
                        isEditing={editingId === profile.id}
                        onEdit={() => setEditingId(profile.id)}
                        onCancelEdit={() => setEditingId(null)}
                        onView={() => handleViewProfile(profile.id)}
                        isLoading={isPending}
                    />
                ))}
            </div>

            {viewingProfile && (
                <ProfileDetailModal
                    profile={viewingProfile.profile}
                    orders={viewingProfile.orders}
                    favorites={viewingProfile.favorites}
                    onClose={() => setViewingProfile(null)}
                />
            )}
        </div>
    );
};

type ProfileCardProps = {
    profile: ProfileRecord;
    isEditing: boolean;
    onEdit: () => void;
    onCancelEdit: () => void;
    onView: () => void;
    isLoading: boolean;
};

const ProfileCard = ({
    profile,
    isEditing,
    onEdit,
    onCancelEdit,
    onView,
    isLoading,
}: ProfileCardProps) => {
    const [state, formAction] = useFormState(updateProfileAction, initialState);

    return (
        <div className="rounded-2xl border border-white/10 bg-yellow-900/40
p-4">
            {isEditing ? (
                <form action={formAction} className="space-y-3">
                    <input type="hidden" name="id" value={profile.id} />
```

```
                    <div className="grid gap-3 md:grid-cols-2">
                        <label className="space-y-2 text-sm">
                            <span className="text-yellow-300">Display
Name</span>
                            <input
                                name="display_name"
                                defaultValue={profile.display_name || ""}
                                className="w-full rounded-lg border border-
white/10 bg-yellow-900/60 px-3 py-2 text-white outline-none focus:border-
indigo-400 focus:ring-2 focus:ring-indigo-400/40"
                            />
                        </label>
                        <label className="space-y-2 text-sm">
                            <span className="text-yellow-300">Phone</span>
                            <input
                                name="phone"
                                defaultValue={profile.phone || ""}
                                className="w-full rounded-lg border border-
white/10 bg-yellow-900/60 px-3 py-2 text-white outline-none focus:border-
indigo-400 focus:ring-2 focus:ring-indigo-400/40"
                            />
                        </label>
                    </div>

                    <label className="block space-y-2 text-sm">
                        <span className="text-yellow-300">Role</span>
                        <select
                            name="role"
                            defaultValue={profile.role}
                            className="w-full rounded-lg border border-white/10
bg-yellow-900/60 px-3 py-2 text-white outline-none focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
                        >
                            <option value="customer">Customer</option>
                            <option value="admin">Admin</option>
                        </select>
                    </label>

                    <label className="block space-y-2 text-sm">
                        <span className="text-yellow-300">Address</span>
                        <input
                            name="address"
                            defaultValue={profile.address || ""}
                            className="w-full rounded-lg border border-white/10
bg-yellow-900/60 px-3 py-2 text-white outline-none focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
                        />
                    </label>

                    <div className="grid gap-3 md:grid-cols-3">
                        <label className="space-y-2 text-sm">
                            <span className="text-yellow-300">City</span>
                            <input
                                name="city"
                                defaultValue={profile.city || ""}
                                className="w-full rounded-lg border border-
white/10 bg-yellow-900/60 px-3 py-2 text-white outline-none focus:border-
indigo-400 focus:ring-2 focus:ring-indigo-400/40"
                            />
                        </label>
                        <label className="space-y-2 text-sm">
                            <span className="text-yellow-300">State</span>
                            <input
                                name="state"
```

```
                                defaultValue={profile.state || ""}
                                className="w-full rounded-lg border border-
white/10 bg-yellow-900/60 px-3 py-2 text-white outline-none focus:border-
indigo-400 focus:ring-2 focus:ring-indigo-400/40"
                            />
                        </label>
                        <label className="space-y-2 text-sm">
                            <span className="text-yellow-300">Zip Code</span>
                            <input
                                name="zip_code"
                                defaultValue={profile.zip_code || ""}
                                className="w-full rounded-lg border border-
white/10 bg-yellow-900/60 px-3 py-2 text-white outline-none focus:border-
indigo-400 focus:ring-2 focus:ring-indigo-400/40"
                            />
                        </label>
                    </div>

                    <label className="block space-y-2 text-sm">
                        <span className="text-yellow-300">Country</span>
                        <input
                            name="country"
                            defaultValue={profile.country || ""}
                            className="w-full rounded-lg border border-white/10
bg-yellow-900/60 px-3 py-2 text-white outline-none focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
                        />
                    </label>

                    {/* Show all profile info in edit mode */}
                    <div className="rounded-lg border border-indigo-500/30 bg-
indigo-500/5 p-3">
                        <p className="mb-2 text-xs font-semibold uppercase
tracking-wider text-indigo-300">
                            Additional Info
                        </p>
                        <div className="grid gap-2 text-sm">

                            <div className="flex justify-between">
                                <span className="text-yellow-400">User
ID:</span>
                                <span className="font-mono text-xs text-
yellow-300">
                                    {profile.uid?.slice(0, 12)}...
                                </span>
                            </div>
                            <div className="flex justify-between">
                                <span className="text-yellow-400">Tier
Status:</span>
                                <span className="font-semibold text-
white">{profile.tier_status}</span>
                            </div>

                        </div>
                    </div>

                    {state.error && <p className="text-sm text-
rose-400">{state.error}</p>}
                    {state.success && (
                        <p className="text-sm text-emerald-400">{state.success}
</p>
                    )}

                    <div className="flex gap-2">
```

```
                        <SubmitButton
                            label="Save"
                            loadingLabel="Saving..."
                            className="bg-indigo-600 px-4 py-2 text-sm"
                        />
                        <button
                            type="button"
                            onClick={onCancelEdit}
                            className="rounded-lg bg-yellow-800 px-4 py-2 text-
sm text-white hover:bg-yellow-700"
                        >
                            Cancel
                        </button>
                    </div>
                </form>
            ) : (
                <>
                    <div className="flex items-start justify-between">
                        <div className="flex-1">
                            <div className="flex items-center gap-3">
                                <h3 className="font-semibold text-white">
                                    {profile.display_name || "No name"}
                                </h3>
                                <span className="rounded-full bg-indigo-500/20
px-2 py-0.5 text-xs font-medium uppercase text-indigo-300">
                                    {profile.role}
                                </span>
                            </div>
                            <p className="text-sm text-
yellow-400">{profile.email}</p>
                            {profile.phone && (
                                <p className="text-sm text-
yellow-500">{profile.phone}</p>
                            )}
                            <p className="mt-1 text-xs text-yellow-600">
                                Last login:{" "}
                                {profile.last_login
                                    ? formatDistanceToNow(new
Date(profile.last_login), {
                                        addSuffix: true,
                                    })
                                    : "Never"}
                            </p>
                        </div>
                        <div className="flex gap-2">
                            <button
                                type="button"
                                onClick={onView}
                                disabled={isLoading}
                                className="flex items-center gap-1 rounded-lg
bg-indigo-600 px-3 py-1 text-xs text-white hover:bg-indigo-700
disabled:opacity-50"
                            >
                                <HiOutlineEye className="h-4 w-4" />
                                View
                            </button>
                            <button
                                type="button"
                                onClick={onEdit}
                                className="rounded-lg bg-yellow-800 px-3 py-1
text-xs text-white hover:bg-yellow-700"
                            >
                                Edit
                            </button>
```

```
                        </div>
                    </div>

                    {profile.address && (
                        <div className="mt-3 text-sm text-yellow-400">
                            <p>{profile.address}</p>
                            <p>
                                {[profile.city, profile.state, profile.zip_code]
                                    .filter(Boolean)
                                    .join(", ")}
                            </p>
                            {profile.country && <p>{profile.country}</p>}
                        </div>
                    )}


                    <div className="mt-3 grid grid-cols-2 gap-2 text-sm">
                        <div>
                            <p className="text-xs text-yellow-500">Tier
Status</p>
                            <p className="font-semibold text-
white">{profile.tier_status}</p>
                        </div>
                    </div>
                </>
            )}


        </div>
    );
};
</file>

<file path="admin/src/lib/data/categories-actions.ts">
"use server";

import { revalidatePath } from "next/cache";
import { createSupabaseServerClient } from "@/lib/supabase/server";

export type CategoryActionState = {
    error?: string;
    success?: string;
};

const sanitize = (value: FormDataEntryValue | null) =>
    String(value ?? "").trim();

export const createCategoryAction = async (
    _prev: CategoryActionState,
    formData: FormData,
): Promise<CategoryActionState> => {
    console.log("createCategoryAction: Received formData");
    const categoryName = sanitize(formData.get("category_name"));
    const image = sanitize(formData.get("image")) || null;
    console.log("createCategoryAction: Image URL:", image);
    const description = sanitize(formData.get("description")) || null;
    const isHidden = formData.get("is_hidden") === "on";

    if (!categoryName) {
        return { error: "Category name is required." };
    }

    const supabase = await createSupabaseServerClient();
```

```
    const { data: { user }, error: authError } = await supabase.auth.getUser();
    console.log("createCategoryAction: User:", user?.id);
    if (authError) {
        console.error("createCategoryAction: Auth Error:", authError);
    }

    if (!user) {
        console.error("createCategoryAction: No authenticated user found");
        return { error: "You must be logged in to perform this action." };
    }

    const categoryData = {
        category_name: categoryName,
        image,
        description,
        is_hidden: isHidden,
    };
    console.log("createCategoryAction: Inserting data:", categoryData);

    const { error } = await supabase
        .from("products_category")
        .insert(categoryData as never);

    if (error) {
        console.error("Failed to create category", error);
        return { error: error.message };
    }

    revalidatePath("/categories");
    revalidatePath("/products");
    return { success: "Category created successfully." };
};

export const updateCategoryAction = async (
    _prev: CategoryActionState,
    formData: FormData,
): Promise<CategoryActionState> => {
    console.log("updateCategoryAction: Received formData");
    const id = sanitize(formData.get("id"));
    const categoryName = sanitize(formData.get("category_name"));
    const image = sanitize(formData.get("image")) || null;
    console.log("updateCategoryAction: Image URL:", image);
    const description = sanitize(formData.get("description")) || null;
    const isHidden = formData.get("is_hidden") === "on";

    if (!id || !categoryName) {
        return { error: "Category ID and name are required." };
    }

    const supabase = await createSupabaseServerClient();
    const updateData = {
        category_name: categoryName,
        image,
        description,
        is_hidden: isHidden,
    };

    const { error } = await supabase
        .from("products_category")
        .update(updateData as never)
        .eq("id", id);

    if (error) {
        console.error("Failed to update category", error);
```

```
        return { error: error.message };
    }

    revalidatePath("/categories");
    revalidatePath("/products");
    return { success: "Category updated successfully." };
};

export const deleteCategoryAction = async (
    categoryId: string,
): Promise<CategoryActionState> => {
    if (!categoryId) {
        return { error: "Category ID is required." };
    }

    const supabase = await createSupabaseServerClient();

    // Check if category has products
    const { data: products } = await supabase
        .from("products")
        .select("id")
        .eq("category_name", categoryId)
        .limit(1);

    if (products && products.length > 0) {
        return { error: "Cannot delete category with existing products." };
    }

    const { error } = await supabase
        .from("products_category")
        .delete()
        .eq("id", categoryId);

    if (error) {
        console.error("Failed to delete category", error);
        return { error: error.message };
    }

    revalidatePath("/categories");
    revalidatePath("/products");
    return { success: "Category deleted successfully." };
};
</file>

<file path="admin/src/lib/data/orders-actions.ts">
"use server";

import { formatISO } from "date-fns";
import { revalidatePath } from "next/cache";
import { createSupabaseServerClient } from "@/lib/supabase/server";
import type { OrderStatus } from "@/lib/types";
import type { Database } from "@/lib/database.types";

export type OrderActionState = {
  error?: string;
  success?: string;
};

const normalizeStatus = (value: string): OrderStatus | null => {
  const candidates: OrderStatus[] = [
    "pending",
    "processing",
    "completed",
    "cancelled",
```

```
  ];
  return candidates.find((candidate) => candidate === value) ?? null;
};

export const updateOrderStatusAction = async (
  _prev: OrderActionState,
  formData: FormData,
): Promise<OrderActionState> => {
  const orderId = String(formData.get("orderId") ?? "").trim();
  const statusValue = String(formData.get("status") ?? "").trim();
  const status = normalizeStatus(statusValue);

  if (!orderId || !status) {
    return { error: "Order ID or status missing." };
  }

  const supabase = await createSupabaseServerClient();
  const updateData: Database["public"]["Tables"]["orders"]["Update"] = {
    status,
    updated_at: formatISO(new Date()),
  };
  const { error } = await supabase
    .from("orders")
    .update(updateData as never)
    .eq("id", orderId);

  if (error) {
    console.error("Failed to update order status", error);
    return { error: error.message };
  }

  revalidatePath("/orders");
  return { success: `Marked as ${status}.` };
};
</file>

<file path="admin/src/lib/database.types.ts">
export type Json = string | number | boolean | null | { [key: string]: Json |
undefined } | Json[]

export type Database = {
  public: {
    Tables: {
      admins: {
        Row: {
          user_id: string
          created_at: string
        }
        Insert: {
          user_id: string
          created_at?: string
        }
        Update: {
          user_id?: string
          created_at?: string
        }
        Relationships: []
      }
      comments: {
        Row: {
          id: string
          product_id: string | null
          user_id: string | null
```

```
      user_name: string | null
      body: string
      created_at: string
    }
    Insert: {
      id?: string
      product_id?: string | null
      user_id?: string | null
      user_name?: string | null
      body: string
      created_at?: string
    }
    Update: {
      id?: string
      product_id?: string | null
      user_id?: string | null
      user_name?: string | null
      body?: string
      created_at?: string
    }
    Relationships: []
  }
  contact: {
    Row: {
      id: string
      name: string
      email: string
      phone: string | null
      message: string
      created_at: string
    }
    Insert: {
      id?: string
      name: string
      email: string
      phone?: string | null
      message: string
      created_at?: string
    }
    Update: {
      id?: string
      name?: string
      email?: string
      phone?: string | null
      message?: string
      created_at?: string
    }
    Relationships: []
  }
  featured_items: {
    Row: {
      id: string
      name: string
      description: string
      image_url: string | null
      likes: string[] | null
      comments: Json | null
      created_at: string
    }
    Insert: {
      id?: string
      name: string
      description: string
      image_url?: string | null
```

```
      likes?: string[] | null
      comments?: Json | null
      created_at?: string
    }
    Update: {
      id?: string
      name?: string
      description?: string
      image_url?: string | null
      likes?: string[] | null
      comments?: Json | null
      created_at?: string
    }
    Relationships: []
  }
  orders: {
    Row: {
      id: string
      user_id: string | null
      items: Json
      total_price: number
      total_quantity: number
      status: string | null
      created_at: string
      updated_at: string | null
    }
    Insert: {
      id?: string
      user_id?: string | null
      items: Json
      total_price: number
      total_quantity: number
      status?: string | null
      created_at?: string
      updated_at?: string | null
    }
    Update: {
      id?: string
      user_id?: string | null
      items?: Json
      total_price?: number
      total_quantity?: number
      status?: string | null
      created_at?: string
      updated_at?: string | null
    }
    Relationships: []
  }
  photo_boot_bookings: {
    Row: {
      id: string
      name: string
      email: string
      phone: string | null
      date: string
      time: string
      package: string
      people: number
      message: string | null
      created_at: string
    }
    Insert: {
      id?: string
      name: string
```

```
      email: string
      phone?: string | null
      date: string
      time: string
      package: string
      people: number
      message?: string | null
      created_at?: string
    }
    Update: {
      id?: string
      name?: string
      email?: string
      phone?: string | null
      date?: string
      time?: string
      package?: string
      people?: number
      message?: string | null
      created_at?: string
    }
    Relationships: []
  }
  products: {
    Row: {
      id: string
      name: string
      slug: string | null
      description: string | null
      price: number | null
      category_name: string | null
      image_url: string | null
      stock: number | null
      likes: string[] | null
      badge: string | null
      created_at: string
      is_hidden: boolean | null
    }
    Insert: {
      id?: string
      name: string
      slug?: string | null
      description?: string | null
      price?: number | null
      category_name?: string | null
      image_url?: string | null
      stock?: number | null
      likes?: string[] | null
      badge?: string | null
      created_at?: string
      is_hidden?: boolean | null
    }
    Update: {
      id?: string
      name?: string
      slug?: string | null
      description?: string | null
      price?: number | null
      category_name?: string | null
      image_url?: string | null
      stock?: number | null
      likes?: string[] | null
      badge?: string | null
      created_at?: string
```

```
      is_hidden?: boolean | null
    }
    Relationships: []
  }
  products_category: {
    Row: {
      id: string
      category_name: string
      image: string | null
      description: string | null
      created_at: string
      is_hidden: boolean | null
    }
    Insert: {
      id?: string
      category_name: string
      image?: string | null
      description?: string | null
      created_at?: string
      is_hidden?: boolean | null
    }
    Update: {
      id?: string
      category_name?: string
      image?: string | null
      description?: string | null
      created_at?: string
      is_hidden?: boolean | null
    }
    Relationships: []
  }
  profiles: {
    Row: {
      id: string
      email: string | null
      display_name: string | null
      phone: string | null
      metadata: Json | null
      created_at: string
      role: string | null
      uid: string | null
      email_verified: boolean | null
      photo_url: string | null
      address: string | null
      city: string | null
      state: string | null
      zip_code: string | null
      country: string | null
      theme: string | null
      tier_status: string | null
      referral_code: string | null
      preferences: Json | null
      saved_payment_methods: Json | null
      updated_at: string | null
      last_login: string | null
    }
    Insert: {
      id?: string
      email?: string | null
      display_name?: string | null
      phone?: string | null
      metadata?: Json | null
      created_at?: string
      role?: string | null
```

```
      uid?: string | null
      email_verified?: boolean | null
      photo_url?: string | null
      address?: string | null
      city?: string | null
      state?: string | null
      zip_code?: string | null
      country?: string | null
      theme?: string | null
      referral_code?: string | null
      preferences?: Json | null
      saved_payment_methods?: Json | null
      updated_at?: string | null
      last_login?: string | null
    }
    Update: {
      id?: string
      email?: string | null
      display_name?: string | null
      phone?: string | null
      metadata?: Json | null
      created_at?: string
      role?: string | null
      uid?: string | null
      email_verified?: boolean | null
      photo_url?: string | null
      address?: string | null
      city?: string | null
      state?: string | null
      zip_code?: string | null
      country?: string | null
      theme?: string | null
      referral_code?: string | null
      preferences?: Json | null
      saved_payment_methods?: Json | null
      updated_at?: string | null
      last_login?: string | null
    }
    Relationships: []
  }
  testimonials: {
    Row: {
      id: string
      text: string
      author: string
      rating: number | null
      likes: string[] | null
      comments: Json | null
      created_at: string
    }
    Insert: {
      id?: string
      text: string
      author: string
      rating?: number | null
      likes?: string[] | null
      comments?: Json | null
      created_at?: string
    }
    Update: {
      id?: string
      text?: string
      author?: string
      rating?: number | null
```

```
          likes?: string[] | null
          comments?: Json | null
          created_at?: string
        }
        Relationships: []
      }
      user_favorites: {
        Row: {
          user_id: string
          product_id: string
          created_at: string | null
        }
        Insert: {
          user_id: string
          product_id: string
          created_at?: string | null
        }
        Update: {
          user_id?: string
          product_id?: string
          created_at?: string | null
        }
        Relationships: []
      }
    }
    Views: {
      [_ in never]: never
    }
    Functions: {
      [_ in never]: never
    }
    Enums: {
      [_ in never]: never
    }
    CompositeTypes: {
      [_ in never]: never
    }
  }
}
</file>

<file path="admin/src/lib/types/products.ts">
export interface NutritionalInfo {
  calories: number;
  protein: string;
  carbs: string;
  fat: string;
  fiber: string;
  sugar: string;
}

export interface Product {
  ProductID: string;
  Name: string;
  Description: string[];
  Price: number;
  Image?: string;
  badge?: string;
  rating?: number;
  nutritional?: NutritionalInfo;
  allergens?: string[];
  isVegetarian?: boolean;
  isSpicy?: boolean;
}
```

```
export type ProductCategory = {
  id: number;
  Name: string;
  Products: Product[];
  Image?: string;
  Description?: string;
}[];

export interface ProductRecord {
  id: string;
  name: string;
  description?: string;
  category_name?: string;
  price: number;
  stock?: number;
  image_url?: string;
  badge?: string;
  created_at?: string;
  is_featured?: boolean;
  likes?: string[]; // UUID array
  is_hidden?: boolean;
}
</file>

<file path="client/package.json">
{
  "name": "client",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "analyze": "cross-env ANALYZE=true next build",
    "start": "next start",
    "lint": "biome check",
    "format": "biome format --write"
  },
  "dependencies": {
    "@radix-ui/react-slot": "^1.2.4",
    "@supabase/auth-helpers-nextjs": "^0.10.0",
    "@supabase/ssr": "^0.7.0",
    "@supabase/supabase-js": "^2.81.1",
    "@vercel/analytics": "^1.5.0",
    "class-variance-authority": "^0.7.1",
    "classnames": "^2.5.1",
    "clsx": "^2.1.1",
    "next": "16.0.1",
    "react": "19.2.0",
    "react-dom": "19.2.0",
    "react-hook-form": "^7.66.0",
    "react-icons": "^5.5.0",
    "tailwind-merge": "^3.4.0"
  },
  "devDependencies": {
    "@biomejs/biome": "2.2.0",
    "@tailwindcss/postcss": "^4",
    "@types/node": "^20",
    "@types/react": "^19",
    "@types/react-dom": "^19",
    "babel-plugin-react-compiler": "1.0.0",
    "cross-env": "^7.0.3",
    "tailwindcss": "^4",
    "tw-animate-css": "^1.4.0",
```

```
    "typescript": "^5"
  }
}
</file>

<file path="client/src/app/about/page.tsx">
import OurJourney from './components/OurJourney';
import OurCommitment from './components/OurCommitment';
import WhatMakesUsSpecial from './components/WhatMakesUsSpecial';
import CoreValues from './components/CoreValues';
import OurVision from './components/OurVision';
import KeyLeadership from './components/KeyLeadership';
import CompanyDescription from './components/CompanyDescription';
import KeyHighlights from './components/KeyHighlights';
import StrengthsAndOpportunities from './components/StrengthsAndOpportunities';
import WeaknessesAndThreats from './components/WeaknessesAndThreats';
import MilestonesAndVision from './components/MilestonesAndVision';
import Main from '@/components/ui/layout/Main';
import { IoFastFoodOutline } from 'react-icons/io5';

const AboutPage = () => {
      return (
            <Main
                  tittle='About PA Luxe Creation'
                  Icon={IoFastFoodOutline}>
                  <OurJourney />
                  <OurVision />
                  <CompanyDescription />
                  <OurCommitment />
                  <MilestonesAndVision />
                  <WhatMakesUsSpecial />
                  <KeyHighlights />
                  <CoreValues />
                  <StrengthsAndOpportunities />
                  <WeaknessesAndThreats />
                  <KeyLeadership />
            </Main>
      );
};

export default AboutPage;
</file>

<file path="client/src/app/gallery/components/ImageGallery.tsx">
'use client';
import React, { useState, useRef, useEffect } from 'react';
import { createClient } from '@/lib/supabase/client';

import Image from 'next/image';
import { FaImages } from 'react-icons/fa';
import dynamic from 'next/dynamic';

const ImageModal = dynamic(() => import('./ImageModal'), { ssr: false, loading:
() => null });
import Section from '@/components/ui/layout/Section';

type ProductRow = {
      id: string;
      name: string;
      description: string | null;
      price: number | null;
      image_url: string | null;
      badge: string | null;
      likes: number | null;
```

```
};

const ImageGallery: React.FC = () => {
	const [modalOpen, setModalOpen] = useState(false);
	const [product, setproduct] = useState<ProductType | null>(null);
	const [allProducts, setAllProducts] = useState<ProductType[]>([]);
	const [loading, setLoading] = useState(true);
	const modalRef = useRef<HTMLDivElement>(null);

	useEffect(() => {
		const fetchProducts = async () => {
			const supabase = createClient();
			const { data, error } = await supabase
				.from('products')
				.select('*');

			if (error) {
				console.error('Error fetching products:', error);
				setLoading(false);
				return;
			}

			const products: ProductType[] = ((data as ProductRow[]) ||
[]).map((p) => ({
				ProductID: p.id,
				Name: p.name,
				Description: [p.description || ''],
				Price: p.price || 0,
				Image: p.image_url || '/Menus/placeholder.png',
				badge: p.badge || undefined,
				rating: p.likes ? 5 : undefined,
			}));

			setAllProducts(products);
			setLoading(false);
		};

		fetchProducts();
	}, []);

	useEffect(() => {
		if (!modalOpen) return;
		const handleKeyDown = (e: KeyboardEvent) => {
			if (e.key === 'Escape') setModalOpen(false);
			if (e.key === 'Tab' && modalRef.current) {
				const focusable =
modalRef.current.querySelectorAll<HTMLElement>(
					'button, [tabindex]:not([tabindex="-1"])'
				);
				if (focusable.length === 0) return;
				const first = focusable[0];
				const last = focusable[focusable.length - 1];
				if (!e.shiftKey && document.activeElement === last) {
					e.preventDefault();
					first.focus();
				} else if (e.shiftKey && document.activeElement ===
first) {
					e.preventDefault();
					last.focus();
				}
			}
		};
		document.addEventListener('keydown', handleKeyDown);
		return () => document.removeEventListener('keydown', handleKeyDown);
```

```
      }, [modalOpen]);

      const handleBackdropClick = (e: React.MouseEvent<HTMLDivElement>) => {
            if (e.target === e.currentTarget) setModalOpen(false);
      };

      const openModal = (product: ProductType) => {
            setproduct(product);
            setModalOpen(true);
      };

      if (loading) {
            return (
                  <Section Icon={FaImages} tittle='Gallery'>
                        <div className='text-center py-10 text-white'>Loading
gallery...</div>
                  </Section>
            );
      }

      return (
            <Section
                  Icon={FaImages}
                  tittle='Gallery'>
                  <div className='grid grid-cols-2 sm:grid-cols-3 md:grid-cols-4
lg:grid-cols-5 gap-4 mt-6'>
                        {allProducts.map(
                              (product) =>
                                    product.Image && (
                                          <article
                                                key={product.Name}
                                                className='group cursor-pointer
rounded-md focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-
offset-black/50 focus:ring-yellow-500
                                                relative w-full aspect-w-1
aspect-h-1 rounded-md overflow-hidden transition-transform duration-300 m-0 p-2
group-hover:scale-105'
                                                tabIndex={0}
                                                aria-label={`Show full image of $
{product.Name}`}
                                                onClick={() =>
openModal(product)}
                                                onKeyDown={(e) => {
                                                      if (e.key === 'Enter' ||
e.key === ' ') {
                                                            e.preventDefault();
                                                            openModal(product);
                                                      }
                                                }}>
                                                <Image
                                                      src={product.Image}
                                                      alt={`${product.Name}
gallery image`}
                                                      className='object-contain
transition-all duration-300 group-hover:brightness-110'
                                                      width={500}
                                                      height={500}
                                                      sizes='(max-width: 640px)
50vw, (max-width: 1024px) 33vw, 20vw'
                                                />
                                                <div className='absolute inset-0
bg-gradient-to-t from-black/70 to-transparent'></div>
                                                <p className='absolute bottom-2
left-2 right-2 truncate text-center bg-black/70 px-2 py-1 rounded-full text-xs
```

```
font-bold text-white'>
                                                {product.Name}
                                          </p>
                                    </article>
                              )
                        )}
                  </div>
                  {modalOpen && product && (
                        <ImageModal
                              product={product}
                              handleBackdropClick={handleBackdropClick}
                              setModalOpen={setModalOpen}
                        />
                  )}
            </Section>
      );
};

export default ImageGallery;
</file>

<file path="client/src/app/menu/[Products]/components/CommentModal.tsx">
import { useState } from 'react';
import { createClient } from '@/lib/supabase/client';

import { useAuth } from '@/lib/supabase/auth/useAuth';
import { useParams, useRouter } from 'next/navigation';
interface CommentModalProps {
      product: ProductType;
      handleCloseCommentModal: () => void;
}
const CommentModal: React.FC<CommentModalProps> = ({ product,
handleCloseCommentModal }) => {
      const [commentText, setCommentText] = useState('');
      const { user } = useAuth();
      const route = useRouter();
      const params = useParams() as { Products: string };
      const handleSubmitComment = async () => {
            if (!product?.Name || !commentText.trim() || !user?.id) {
                  handleCloseCommentModal();
                  return;
            }
            try {
                  const supabase = createClient();
                  // @ts-ignore - Supabase type inference issue with Database
types
                  await supabase.from('comments').insert({
                        product_id: product.ProductID || product.Name,
                        user_id: user?.id || null,
                        user_name: (user as any)?.user_metadata?.full_name ||
(user as any)?.email || 'Anonymous',
                        body: commentText.trim(),
                        created_at: new Date().toISOString(),
                  });
                  setCommentText('');
                  handleCloseCommentModal();
                  route.push(`/menu/${params.Products}/$
{product.Name.toLowerCase().replace(/\s+/g, '-')}`);
            } catch (error) {
                  console.error('Error adding comment:', error);
                  alert('Failed to add comment. Please try again.');
            }
      };
      return (
```

```
            <main className='fixed inset-0 bg-black flex items-center justify-
center z-50 p-4'>
                  <div className='bg-black/50 p-6 rounded-md shadow-lg w-full
max-w-md text-white'>
                        <h2>Add a Comment</h2>
                        <textarea
                              className='w-full p-3 bg-transparent border
border-white rounded-md mb-4 focus:outline-none focus:ring-2 focus:ring-white
text-white placeholder-white/70'
                              rows={4}
                              placeholder='Enter your comment...'
                              value={commentText}
                              onChange={(e) => setCommentText(e.target.value)}
></textarea>
                        <div className='flex justify-end gap-2'>
                              <button
                                    className='px-4 py-2 bg-yellow-700 rounded-
md hover:bg-yellow-600 focus:outline-none focus:ring-2 focus:ring-white'
                                    onClick={handleCloseCommentModal}>
                                    Cancel
                              </button>
                              <button
                                    className='px-4 py-2 bg-yellow-700 text-
white rounded-md hover:bg-yellow-800 focus:outline-none focus:ring-2 focus:ring-
white'
                                    onClick={handleSubmitComment}>
                                    Submit Comment
                              </button>
                        </div>
                  </div>
            </main>
      );
};

export default CommentModal;
</file>

<file path="client/src/app/menu/[Products]/components/SocialButtons.tsx">
'use client';

import { BiSolidShare } from 'react-icons/bi';
import LikesButton from './LikesButton';

import CommentsButton from './CommentsButton';

interface SocialButtonsProps {
      product: ProductType;
}

const SocialButtons: React.FC<SocialButtonsProps> = ({ product }) => {
      const handleShare = () => {
            if (navigator.share) {
                  navigator
                        .share({
                              title: product.Name,
                              text: `Check out this product: ${product.Name}`,
                              url: window.location.href, // Or the specific
product URL
                        })
                        .catch((error) => console.error('Error sharing:',
error));
            } else {
                  // Fallback for browsers that don't support Web Share API
                  alert('Sharing is not supported in this browser.');
```

```
                // You could also provide a copy-to-clipboard option here
            }
        };

        return (
            <div className='grid grid-cols-3 w-full border-t border-white/20
pt-4 bg-white/5 -mx-6 px-6 pb-6 -mb-6 rounded-b-xl gap-2'>
                <LikesButton itemId={product.ProductID} table="products" />
                <CommentsButton product={product} />
                {/* Share Button */}
                <span
                    className='flex items-center justify-center gap-1
truncate text-center py-0.5 cursor-pointer text-white font-bold'
                    onClick={handleShare}>
                    <BiSolidShare size={20} />
                    <span className='text-xs text-center
truncate'>Share</span>
                </span>
            </div>
        );
};

export default SocialButtons;
</file>

<file path="client/src/app/menu/components/MenuSections.tsx">
'use client';

import React from 'react';

import CategoryCard from '@/app/menu/components/CategoryCard';
import NoResults from '@/app/menu/components/NoResults';
import ProductCard from './ProductCard';

interface MenuSectionsProps {
    categories: ProductsType;
    matchingProducts: ProductType[];
    productCategoryMap?: Map<string, string>; // Map of ProductID -> category
slug
    isSearching: boolean;
}

const MenuSections: React.FC<MenuSectionsProps> = ({ categories,
matchingProducts, productCategoryMap, isSearching }) => {
    const showSearchResults = isSearching;

    return (
        <>
            {showSearchResults ? (
                <div className='flex flex-col gap-6'>
                    {categories.length > 0 || matchingProducts.length
> 0 ? (
                        <div className='flex flex-col gap-6'>
                            {/* Categories Section */}
                            {categories.length > 0 && (
                                <div className='space-y-4'>
                                    <h2 className='text-2xl
font-bold text-white border-b border-yellow-500 pb-2'>
                                        Matching Categories
                                    </h2>
                                    <div className='grid grid-
cols-1 md:grid-cols-2 lg:grid-cols-4 gap-4'>

    {categories.map((category) => (
```

```
                                                                  <CategoryCard
      key={category.Name}
      group={category}
                                                                  />
                                                            ))}
                                                      </div>
                                                </div>
                                          )}

                                          {/* Products Section */}
                                          {matchingProducts.length > 0 && (
                                                <div className='space-y-4'>
                                                      <h2 className='text-2xl
font-bold text-white border-b border-yellow-500 pb-2'>
                                                            Matching Products
                                                      </h2>
                                                      <div className='grid grid-
cols-1 md:grid-cols-2 lg:grid-cols-4 gap-4'>
      {matchingProducts.map((product) => {
                                                            // Get category
slug from map, fallback to 'menu' if not found
                                                            const
categorySlug = productCategoryMap?.get(product.ProductID) || 'menu';
                                                            return (
      <ProductCard
      key={product.ProductID}
      product={product}
      categoryName={categorySlug}
                                                                  />
                                                            );
                                                      })}
                                                </div>
                                          </div>
                                          )}
                                    </div>
                              ) : (
                                    <NoResults />
                              )}
                        </div>
                  ) : (
                        // Default: show all categories
                        <ul className='grid grid-cols-1 md:grid-cols-2 lg:grid-
cols-4 gap-4'>
                              {categories.map((group: ProductsType[number]) => {
                                    return (
                                          <CategoryCard
                                                key={group.Name}
                                                group={group}
                                          />
                                    );
                              })}
                        </ul>
                  )}
            </>
      );
};
```

```
export default MenuSections;
</file>

<file path="client/src/app/menu/components/SpecialOffers.tsx">
import Section from '@/components/ui/layout/Section';

const SpecialOffers: React.FC = () => {
      return (
            <Section tittle=' Special Offers'>
                  <article>
                        <div className='grid grid-cols-1 md:grid-cols-2 gap-6'>
                              {/* Offer 1 */}
                              <div className='bg-black/50 border border-
yellow-500 rounded-md p-2 flex flex-col'>
                                    <div className='bg-yellow-700 rounded-md p-6
flex flex-col h-full'>
                                          <h3
                                                className='text-xl font-bold
text-white mb-2'
                                                style={{ filter: 'drop-shadow(0 0
2px #f0b100)' }}>
                                                Weekend Promotion
                                          </h3>
                                          <p className='text-white flex-grow'>
                                                Get a Free Pork Plate with Every
Photo boot on Saturdays and Sundays!
                                          </p>
                                    </div>
                              </div>
                              {/* Offer 2 */}
                              <div className='bg-black/50 border border-
yellow-500 rounded-md p-2 flex flex-col'>
                                    <div className='bg-yellow-700 rounded-md p-6
flex flex-col h-full'>
                                          <h3
                                                className='text-xl font-bold
text-white mb-2'
                                                style={{ filter: 'drop-shadow(0 0
2px #f0b100)' }}>
                                                Taxi Partnerships
                                          </h3>
                                          <p className='text-white flex-grow'>
                                                Bulk discounts available for taxi
associations. For example, 10 washes = 1 free!
                                          </p>
                                    </div>
                              </div>
                        </div>
                  </article>
            </Section>
      );
};

export default SpecialOffers;
</file>

<file path="client/src/app/menu/layout.tsx">
'use client';

import { CartProvider } from '@/lib/context/CartContext';
import { ToastProvider } from '@/lib/context/CartContext';

import React, { ReactNode } from 'react';
```

```
interface LayoutProps {
      children: ReactNode;
}

const Layout: React.FC<LayoutProps> = ({ children }) => {
      return (
            <ToastProvider>
                  <CartProvider>{children}</CartProvider>
            </ToastProvider>
      );
};

export default Layout;
</file>

<file path="client/src/app/privacy/page.tsx">
import Main from '@/components/ui/layout/Main';
import Section from '@/components/ui/layout/Section';

const PrivacyPolicyPage: React.FC = () => {
      return (
            <Main tittle='Privacy Policy'>
                  <Section>
                        <div className='space-y-6'>
                              <p>Effective Date: 26 July 2024</p>

                              <article>
                                    <h2 className='text-2xl font-semibold
mb-2'>1. Introduction</h2>
                                    <p>
                                          Welcome to PA Luxe Creation PTY Ltd.
We are committed to protecting your privacy and
                                          ensuring that your personal
information is handled in a safe and responsible manner.
                                          This Privacy Policy outlines how we
collect, use, disclose, and safeguard your
                                          information when you use our services,
including our website and digital booking
                                          system.
                                    </p>
                              </article>

                              <article>
                                    <h2 className='text-2xl font-semibold
mb-2'>2. Information We Collect</h2>
                                    <p>
                                          We may collect information about you
in a variety of ways. The information we may
                                          collect includes:
                                    </p>
                                    <h3 className='text-xl font-semibold mt-4
mb-2'>Personal Data</h3>
                                    <p>
                                          Personally identifiable information,
such as your name, email address, and telephone
                                          number, that you voluntarily give to
us when you use our digital booking system or
                                          contact us.
                                    </p>
                                    <h3 className='text-xl font-semibold mt-4
mb-2'>Usage Data</h3>
                                    <p>
                                          Information that our servers
automatically collect when you access our website, such
```

```
                                        as your IP address, browser type,
operating system, access times, and the pages you
                                        have viewed directly before and after
accessing the site.
                                  </p>
                            </article>

                            <article>
                                  <h2 className='text-2xl font-semibold
mb-2'>3. How We Use Your Information</h2>
                                  <p>
                                        Having accurate information about you
permits us to provide you with a smooth,
                                        efficient, and customized experience.
Specifically, we may use information collected
                                        about you to:
                                  </p>
                                  <ul className='list-disc list-inside ml-4
mt-2'>
                                        <li>Manage your bookings and
appointments for our Photo boot services.</li>
                                        <li>Send you confirmations, reminders,
and updates regarding your bookings.</li>
                                        <li>Respond to your comments and
questions and provide customer service.</li>
                                        <li>
                                              Monitor and analyze usage and
trends to improve your experience with our services.
                                        </li>
                                        <li>Notify you of updates to our
services.</li>
                                  </ul>
                            </article>

                            <article>
                                  <h2 className='text-2xl font-semibold
mb-2'>4. Data Sharing and Disclosure</h2>
                                  <p>
                                        We do not share, sell, rent, or trade
your personal information with third parties for
                                        their commercial purposes. We may
share information we have collected about you in
                                        certain situations, such as:
                                  </p>
                                  <ul className='list-disc list-inside ml-4
mt-2'>
                                        <li>
                                              <strong>By Law or to Protect
Rights:</strong> If we believe the release of
                                              information about you is
necessary to respond to legal process, to investigate or
                                              remedy potential violations of
our policies, or to protect the rights, property, and
                                              safety of others, we may share
your information as permitted or required by any
                                              applicable law, rule, or
regulation.
                                        </li>
                                        <li>
                                              <strong>Service
Providers:</strong> We may share your information with third parties
                                              that perform services for us or
on our behalf, including payment processing, data
                                              analysis, email delivery, hosting
```

```
services, and customer service.
                                    </li>
                              </ul>
                        </article>

                        <article>
                              <h2 className='text-2xl font-semibold
mb-2'>5. Data Security</h2>
                              <p>
                                    We use administrative, technical, and
physical security measures to help protect your
                                    personal information. While we have
taken reasonable steps to secure the personal
                                    information you provide to us, please
be aware that despite our efforts, no security
                                    measures are perfect or impenetrable,
and no method of data transmission can be
                                    guaranteed against any interception or
other type of misuse.
                              </p>
                        </article>

                        <article>
                              <h2 className='text-2xl font-semibold
mb-2'>6. Your Rights</h2>
                              <p>
                                    You have the right to access, correct,
or delete your personal information. You may
                                    also have the right to object to or
restrict certain processing of your data. To
                                    exercise these rights, please contact
us using the contact information provided below.
                              </p>
                        </article>

                        <article>
                              <h2 className='text-2xl font-semibold
mb-2'>7. Changes to This Privacy Policy</h2>
                              <p>
                                    We may update this Privacy Policy from
time to time in order to reflect, for example,
                                    changes to our practices or for other
operational, legal, or regulatory reasons. We
                                    will notify you of any changes by
posting the new Privacy Policy on this page.
                              </p>
                        </article>

                        <article>
                              <h2 className='text-2xl font-semibold
mb-2'>8. Contact Us</h2>
                              <p>
                                    If you have questions or comments
about this Privacy Policy, please contact us at:
                                    <br />
                                    PA Luxe Creation PTY Ltd
                                    <br />
                                    [Your Address Here]
                                    <br />
                                    Email: [Your Email Here]
                                    <br />
                                    Phone: [Your Phone Number Here]
                              </p>
                        </article>
```

```
                    </div>
                </Section>
            </Main>
        );
};
export default PrivacyPolicyPage;
</file>

<file path="client/src/components/Navbar/NavbarClient.tsx">
'use client';
import React from 'react';
import DesktopNavbar from './DesktopNavbar';
import MobileNavbar from './MobileNavbar';
import ProfileMenu from './ProfileMenu';
import AuthButton from './AuthButton';
import MobileMenu from './MobileMenu';
import { useMenubarToggle } from './useMenubarToggle';
import { User } from '@supabase/supabase-js';

interface NavbarClientProps {
    user: User | null;
}

const NavbarClient: React.FC<NavbarClientProps> = ({ user }) => {
    const { profileOpen, mobileOpen, setMenubar } = useMenubarToggle();

    return (
        <header className='sticky top-0 left-0 m-0 p-0 w-full px-4 sm:px-6
lg:px-8 z-50 bg-gradient-to-br from-yellow-900/20 to-amber-900/20 backdrop-blur-
md border-b border-amber-400/10 shadow-lg'>
            <div className='max-w-7xl mx-auto flex justify-between items-center
py-3'>
                <MobileNavbar
                    setMenubar={setMenubar}
                    mobileOpen={mobileOpen}
                />
                <DesktopNavbar />
                <AuthButton setMenubar={setMenubar} user={user} />
            </div>
            {profileOpen ? <ProfileMenu setMenubar={setMenubar} user={user} /> :
null}
            {mobileOpen ? <MobileMenu setMenubar={setMenubar} /> : null}
        </header>
    );
};
export default NavbarClient;
</file>

<file path="client/src/components/ui/AddOnSelector.tsx">
import React from 'react';

const photoBoothAddOns = [
      { id: 'premium-backdrop', label: 'Premium Backdrop', price: '150' },
      { id: 'extra-prints', label: 'Extra Prints Package', price: '100' },
      { id: 'props-upgrade', label: 'Props Upgrade', price: '80' },
      { id: 'digital-album', label: 'Digital Album', price: '50' },
];

interface AddOnSelectorProps {
      selectedAddOns: string[];
      onChange: (addOns: string[]) => void;
}

const AddOnSelector: React.FC<AddOnSelectorProps> = ({ selectedAddOns,
```

```
onChange }) => {
      const handleAddOnChange = (addOn: string) => {
            const newSelection = selectedAddOns.includes(addOn)
                  ? selectedAddOns.filter((item) => item !== addOn)
                  : [...selectedAddOns, addOn];
            onChange(newSelection);
      };

      return (
            <div className='space-y-4'>
                  <h3 className='text-lg font-semibold text-white'>Additional
Services</h3>
                  <div className='grid grid-cols-1 md:grid-cols-2 gap-4'>
                        {photoBoothAddOns.map(({ id, label, price }) => (
                              <div
                                    key={id}
                                    className='flex items-center space-x-3 p-3
rounded-md bg-black/30 border border-white/20'>
                                    <input
                                          type='checkbox'
                                          id={id}

      checked={selectedAddOns.includes(label)}
                                          onChange={() =>
handleAddOnChange(label)}
                                          className='h-5 w-5 text-yellow-500
rounded focus:ring-yellow-500 bg-black/30 border-white/20'
                                    />
                                    <label
                                          htmlFor={id}
                                          className='flex-1 cursor-pointer'>
                                          <span className='block text-
white'>{label}</span>
                                          <span className='block text-sm text-
white/60'>R{price}</span>
                                    </label>
                              </div>
                        ))}
                  </div>
            </div>
      );
};

export default AddOnSelector;
</file>

<file path="client/src/components/ui/Button.tsx">
import React from 'react';
import classNames from 'classnames';

export type ButtonVariant = 'primary' | 'secondary' | 'danger' | 'icon' |
'suggestion';
export type ButtonSize = 'sm' | 'md' | 'lg';

interface ButtonProps extends React.ButtonHTMLAttributes<HTMLButtonElement> {
      variant?: ButtonVariant;
      size?: ButtonSize;
      loading?: boolean;
}

const baseClasses =
      'rounded-md flex gap-2 items-center justify-center font-semibold
transition-colors duration-200 focus:outline-none focus-visible:ring-4  focus-
visible:ring-yellow-400 text-nowrap';
```

```
const variantClasses = {
      primary: 'bg-yellow-600 text-white hover:bg-yellow-700 shadow-sm shadow-
black',
      secondary: 'bg-white text-yellow-600 border border-yellow-400 hover:bg-
red-50',
      danger: 'bg-yellow-100 text-yellow-700 border border-yellow-400 hover:bg-
yellow-200',
      icon: 'bg-none text-yellow-500 hover:text-yellow-700',
      suggestion: 'bg-transparent text-white hover:bg-yellow-700',
};
const sizeClasses = {
      sm: 'px-3 py-1 text-sm',
      md: 'px-4 py-2 text-base',
      lg: 'px-6 py-3 text-lg',
};

const Button: React.FC<ButtonProps> = ({
      children,
      variant = 'primary',
      size = 'sm',
      loading = false,
      disabled,
      className,
      ...props
}) => (
      <button
            className={classNames(baseClasses, variantClasses[variant],
sizeClasses[size], className, {
                  'opacity-50 cursor-not-allowed': disabled || loading,
            })}
            disabled={disabled || loading}
            {...props}>
            {loading ? (
                  <span className='inline-block w-4 h-4 border-2 border-white
border-t-yellow-600 rounded-full animate-spin mr-2'></span>
            ) : (
                  children
            )}
      </button>
);

export default Button;
</file>

<file path="client/src/components/ui/layout/FormField.tsx">
import React from 'react';

interface FormFieldProps {
      label: string;
      name: string;
      type?: 'text' | 'date' | 'time' | 'select';
      value?: string;
      onChange?: (e: React.ChangeEvent<HTMLInputElement | HTMLSelectElement>) =>
void;
      required?: boolean;
      placeholder?: string;
      className?: string;
      children?: React.ReactNode;
      min?: string;
      id?: string;
}

const formInputStyle =
      'w-full p-3 rounded-md bg-black/30 border border-white/20 focus:ring-2
```

```
focus:ring-yellow-500 focus:border-yellow-500 outline-none';

const FormField: React.FC<FormFieldProps> = ({
      label,
      name,
      type = 'text',
      value,
      onChange,
      required = false,
      placeholder,
      className = '',
      children,
      min,
      id,
}) => (
      <div className={className}>
            <label
                  htmlFor={id || name}
                  className='block mb-2 font-semibold text-white'>
                  {label}
            </label>
            {type === 'select' ? (
                  <select
                        name={name}
                        id={id || name}
                        value={value}
                        onChange={onChange}
                        required={required}
                        className={formInputStyle}>
                        {children}
                  </select>
            ) : (
                  <input
                        type={type}
                        name={name}
                        id={id || name}
                        value={value}
                        onChange={onChange}
                        required={required}
                        placeholder={placeholder}
                        className={formInputStyle}
                        min={min}
                  />
            )}
      </div>
);

export default FormField;
</file>

<file path="client/src/components/ui/Link.tsx">
import React from 'react';
import Link, { LinkProps as NextLinkProps } from 'next/link';
import classNames from 'classnames';

export type LinkVariant = 'primary' | 'secondary' | 'button';

interface LinkProps extends NextLinkProps {
      children: React.ReactNode;
      className?: string;
      variant?: LinkVariant;
}

const baseClasses = 'transition-colors duration-200 flex gap-2 items-center
```

```
justify-center';

const variantClasses = {
      primary: 'text-yellow-100 shadow-blue-500 font-bold hover:text-yellow-700
hover:underline',
      secondary: 'text-yellow-200 hover:text-yellow-700 hover:underline',
      button:
            'bg-yellow-600 font-bold grow p-2 text-center text-nowrap text-white
shadow-md shadow-black active:bg-yellow-900 rounded-md flex gap-2 items-center
justify-center font-semibold transition-colors duration-200 focus:outline-none
focus-visible:ring-4  focus-visible:ring-yellow-400 ',
};

const AppLink: React.FC<LinkProps> = ({ children, className, variant =
'primary', ...props }) => {
      return (
            <Link
                  {...props}
                  className={classNames(baseClasses, variantClasses[variant],
className)}>
                  {children}
            </Link>
      );
};

export default AppLink;
</file>

<file path="client/src/lib/forms/ContactForm.tsx">
'use client';

import { useForm } from 'react-hook-form';
import { useState } from 'react';
import { FaWhatsapp, FaEnvelope, FaPhone } from 'react-icons/fa';
import { createClient } from '@/lib/supabase/client';
import { Database } from '@/lib/types/database.types';

type FormData = {
  name: string;
  email: string;
  phone: string;
  message: string;
};

export default function ContactForm() {
  const [isSubmitting, setIsSubmitting] = useState(false);
  const [submitStatus, setSubmitStatus] = useState<{ success: boolean; message:
string } | null>(null);

  const { register, handleSubmit, formState: { errors }, reset } =
useForm<FormData>({
  });

  const onSubmit = async (data: FormData) => {
    setIsSubmitting(true);
    try {
      const supabase = createClient();
      const insertData: Database['public']['Tables']['contact']['Insert'] = {
        name: data.name,
        email: data.email,
        phone: data.phone,
        message: data.message
      };
      const { error } = await supabase.from('contact').insert(insertData as
```

```
any);

      if (error) throw error;

      setSubmitStatus({
        success: true,
        message: 'Your message has been sent successfully! We\'ll get back to
you soon.'
      });
      reset();
    } catch (error) {
      console.error('Error submitting contact form:', error);
      setSubmitStatus({
        success: false,
        message: 'Failed to send message. Please try again later.'
      });
    } finally {
      setIsSubmitting(false);
    }
  };

  const openWhatsApp = () => {
    const phoneNumber = '1234567890'; // Replace with your WhatsApp number
    const message = 'Hello, I have a question about your services.';
    window.open(`https://wa.me/${phoneNumber}?text=$
{encodeURIComponent(message)}`, '_blank');
  };

  return (
    <div className="max-w-4xl mx-auto p-6 bg-white/10 backdrop-blur-md rounded-
lg shadow-lg">
      <h2 className="text-3xl font-bold mb-6 text-center text-white">Contact
Us</h2>

      <div className="grid md:grid-cols-2 gap-8">
        <div>
          <h3 className="text-xl font-semibold mb-4 text-white">Get in
Touch</h3>
          <p className="mb-6 text-gray-200">
            Have questions or feedback? We'd love to hear from you. Fill out the
form or reach out directly.
          </p>

          <div className="space-y-4">
            <div className="flex items-center space-x-3">
              <FaPhone className="text-yellow-500/95 text-xl" />
              <span className="text-white">+1 (555) 123-4567</span>
            </div>
            <div className="flex items-center space-x-3">
              <FaEnvelope className="text-yellow-500/95 text-xl" />
              <span className="text-white">info@centraleatery.com</span>
            </div>
            <button
              onClick={openWhatsApp}
              className="flex items-center space-x-2 bg-green-600 hover:bg-
green-700 text-white px-4 py-2 rounded-md transition-colors"
            >
              <FaWhatsapp className="text-xl" />
              <span>Chat on WhatsApp</span>
            </button>
          </div>
        </div>

        <form onSubmit={handleSubmit(onSubmit)} className="space-y-4">
```

```
          {submitStatus && (
            <div className={`p-4 rounded-md ${submitStatus.success ? 'bg-
green-500/20' : 'bg-yellow-500/20'}`}>
              <p className={submitStatus.success ? 'text-green-200' : 'text-
yellow-200'}>
                {submitStatus.message}
              </p>
            </div>
          )}

          <div>
            <label htmlFor="name" className="block text-sm font-medium text-
white mb-1">Name</label>
            <input
              id="name"
              type="text"
              {...register('name')}
              className="w-full px-4 py-2 rounded-md bg-white/10 border border-
white/20 text-white placeholder-gray-400 focus:outline-none focus:ring-2
focus:ring-yellow-500/50"
              placeholder="Your name"
            />
            {errors.name && <p className="mt-1 text-sm text-
yellow-300">{errors.name.message}</p>}
          </div>

          <div>
            <label htmlFor="email" className="block text-sm font-medium text-
white mb-1">Email</label>
            <input
              id="email"
              type="email"
              {...register('email')}
              className="w-full px-4 py-2 rounded-md bg-white/10 border border-
white/20 text-white placeholder-gray-400 focus:outline-none focus:ring-2
focus:ring-yellow-500/50"
              placeholder="your.email@example.com"
            />
            {errors.email && <p className="mt-1 text-sm text-
yellow-300">{errors.email.message}</p>}
          </div>

          <div>
            <label htmlFor="phone" className="block text-sm font-medium text-
white mb-1">Phone</label>
            <input
              id="phone"
              type="tel"
              {...register('phone')}
              className="w-full px-4 py-2 rounded-md bg-white/10 border border-
white/20 text-white placeholder-gray-400 focus:outline-none focus:ring-2
focus:ring-yellow-500/50"
              placeholder="+1 (555) 123-4567"
            />
            {errors.phone && <p className="mt-1 text-sm text-
yellow-300">{errors.phone.message}</p>}
          </div>

          <div>
            <label htmlFor="message" className="block text-sm font-medium text-
white mb-1">Message</label>
            <textarea
              id="message"
              rows={4}
```

```
              {...register('message')}
              className="w-full px-4 py-2 rounded-md bg-white/10 border border-
white/20 text-white placeholder-gray-400 focus:outline-none focus:ring-2
focus:ring-yellow-500/50"
              placeholder="Your message..."
            />
            {errors.message && <p className="mt-1 text-sm text-
yellow-300">{errors.message.message}</p>}
          </div>

          <button
            type="submit"
            disabled={isSubmitting}
            className="w-full bg-yellow-500/95 hover:bg-yellow-600 text-white
font-medium py-2 px-4 rounded-md transition-colors disabled:opacity-70
disabled:cursor-not-allowed"
          >
            {isSubmitting ? 'Sending...' : 'Send Message'}
          </button>
        </form>
      </div>
    </div>
  );
}
</file>

<file path="client/src/lib/supabase/client.ts">
import { createBrowserClient } from '@supabase/ssr';
import { Database } from '@/lib/types/database.types';

export function createClient() {
  return createBrowserClient<Database>(
    process.env.NEXT_PUBLIC_SUPABASE_URL!,
    process.env.NEXT_PUBLIC_SUPABASE_PUBLISHABLE_KEY!
  );
}
</file>

<file path="client/src/lib/supabase/comments.ts">
import { createClient } from './client';


export const getCommentsForProduct = async (productId: string) => {
  const supabaseBrowser = createClient();
  const { data, error } = await supabaseBrowser
    .from('comments')
    .select('*')
    .eq('product_id', productId)
    .order('created_at', { ascending: false });
  if (error) throw error;
  return data;
};

export const addComment = async (productId: string, userId: string | null,
userName: string | null, body: string) => {
  const supabaseBrowser = createClient();
  const { data, error } = await supabaseBrowser
    .from('comments')
    .insert({ product_id: productId, user_id: userId, user_name: userName,
body } as any)
    .select()
    .single();
  if (error) throw error;
  return data;
```

```
};
</file>

<file path="client/src/lib/supabase/favorites.ts">
import { createClient } from './client';

export const getUserFavorites = async (userId: string) => {
  const supabaseBrowser = createClient();
  const { data, error } = await supabaseBrowser
    .from('user_favorites')
    .select('product_id')
    .eq('user_id', userId);
  if (error) throw error;
  return data?.map((r: any) => r.product_id) ?? [];
};

export const toggleFavorite = async (userId: string, productId: string) => {
  const supabaseBrowser = createClient();
  // Try insert; if conflict, delete
  const { error: insertErr } = await supabaseBrowser
    .from('user_favorites')
    .insert([{ user_id: userId, product_id: productId }] as any);
  if (!insertErr) return { added: true };
  // If insert failed because exists, remove
  const { data, error } = await supabaseBrowser
    .from('user_favorites')
    .delete()
    .match({ user_id: userId, product_id: productId });
  if (error) throw error;
  return { added: false };
};
</file>

<file path="client/src/lib/supabase/orders/orders.ts">
import { createClient } from '@/lib/supabase/client';

export interface Order {
  id: string;
  user_id: string;
  items: CartItem[];
  total_price: number;
  total_quantity: number;
  status: 'pending' | 'processing' | 'completed' | 'cancelled';
  created_at: string;
  updated_at: string;
}

export const addOrder = async (
  userId: string,
  items: CartItem[],
  totalPrice: number,
  totalQuantity: number
) => {
  const supabase = createClient();

  const { data, error } = await supabase
    .from('orders')
    .insert({
      user_id: userId,
      items,
      total_price: totalPrice,
      total_quantity: totalQuantity,
      status: 'pending',
    } as any)
```

```
    .select()
    .single();

  if (error) {
    console.error('Error creating order:', error);
    throw new Error(error.message);
  }

  return data as Order;
};

export const getOrdersByUser = async (userId: string) => {
  const supabase = createClient();

  const { data, error } = await supabase
    .from('orders')
    .select('*')
    .eq('user_id', userId)
    .order('created_at', { ascending: false });

  if (error) {
    console.error('Error fetching orders:', error);
    return [];
  }

  return (data as Order[]) || [];
};

export const getOrdersByStatus = async (userId: string, status: string) => {
  const supabase = createClient();

  const { data, error } = await supabase
    .from('orders')
    .select('*')
    .eq('user_id', userId)
    .eq('status', status)
    .order('created_at', { ascending: false });

  if (error) {
    console.error('Error fetching orders:', error);
    return [];
  }

  return (data as Order[]) || [];
};

export const updateOrderStatus = async (orderId: string, status: string) => {
  const supabase = createClient();

  const { data, error } = await supabase
    .from('orders')
    .update({ status, updated_at: new Date().toISOString() } as never)
    .eq('id', orderId)
    .select()
    .single();

  if (error) {
    console.error('Error updating order:', error);
    throw new Error(error.message);
  }

  return data as Order;
};
</file>
```

<file path="client/src/lib/types/UserTypes.d.ts">
```
import { Reward, yellowemption, paymentMethod } from './index';
declare global {
  type UserRole =
    | 'coFounder'
    | 'Ceo'
    | 'Manager'
    | 'kitchenStaff'
    | 'Cashier'
    | 'Customer'
    | 'admin';

  interface UserPreferences {
    dietaryRestrictions: string[];
    favoriteItems: string[];
    preferyellowPaymentMethod: paymentMethod
    communicationPreferences: {
      email: boolean;
      sms: boolean;
      promotions: boolean;
    };
  }

  interface UserProfile {
    uid: string;
    displayName: string | null;
    email: string | null;
    emailVerified: boolean;
    photoURL: string | null;
    phoneNumber: string | null;
    role: UserRole;
    address: string;
    city: string;
    state: string;
    zipCode: string;
    country: string;
    theme: 'system' | 'light' | 'dark';
    tierStatus: string;
    referralCode: string;
    preferences: UserPreferences;
    savedPaymentMethods?: PaymentMethod[];
    createdAt: Date;
    updatedAt: Date;
    lastLogin?: Date;
  }
}
export { };
```
</file>

<file path="client/tsconfig.json">
```
{
  "compilerOptions": {
    "target": "ES2018",
    "lib": [
      "dom",
      "dom.iterable",
      "esnext"
    ],
    "allowJs": true,
    "skipLibCheck": true,
    "strict": true,
    "noEmit": true,
    "esModuleInterop": true,
```
</file>

```
    "module": "esnext",
    "moduleResolution": "bundler",
    "resolveJsonModule": true,
    "isolatedModules": true,
    "jsx": "react-jsx",
    "incremental": true,
    "plugins": [
      {
        "name": "next"
      }
    ],
    "paths": {
      "@/*": [
        "./src/*"
      ]
    }
  },
  "include": [
    "next-env.d.ts",
    "**/*.ts",
    "**/*.tsx",
    "**/*.mts",
    ".next/types/**/*.ts",
    ".next/dev/types/**/*.ts"
  ],
  "exclude": [
    "node_modules",
    ".next"
  ]
}
</file>

<file path="admin/src/components/layout/nav-items.tsx">
export type NavItem = {
  label: string;
  href: string;
  icon: string;
  badge?: string;
};

export const NAV_ITEMS: NavItem[] = [
  {
    label: "Dashboard",
    href: "/dashboard",
    icon: "HiOutlineChartBar",
  },
  {
    label: "Menu",
    href: "/menu",
    icon: "PiBowlFoodBold",
  },
  {
    label: "Orders",
    href: "/orders",
    icon: "HiOutlineClipboardDocumentList",
    badge: "Live",
  },
  {
    label: "Contact",
    href: "/contact",
    icon: "HiOutlineEnvelope",
  },
  {
    label: "Profiles",
```

```
    href: "/profiles",
    icon: "HiOutlineUsers",
  },
  {
    label: "Photo Bookings",
    href: "/photo-bookings",
    icon: "HiOutlineCamera",
  },
  {
    label: "Testimonials",
    href: "/testimonials",
    icon: "HiOutlineStar",
  },
  {
    label: "Featured Items",
    href: "/featured-items",
    icon: "HiOutlineSparkles",
  },
  {
    label: "Settings",
    href: "/settings",
    icon: "HiOutlineCog",
  },
];
</file>

<file path="admin/src/components/menu/CategoriesBoard.tsx">
"use client";

import { useState } from "react";
import { useFormState } from "react-dom";
import Image from "next/image";
import { HiOutlinePlus, HiChevronDown, HiChevronUp } from "react-icons/hi2";
import { SubmitButton } from "@/components/forms/SubmitButton";
import { DeleteConfirmDialog } from "@/components/shared/DeleteConfirmDialog";
import { ImageUpload } from "@/components/forms/ImageUpload";
import {
    type CategoryActionState,
    createCategoryAction,
    updateCategoryAction,
    deleteCategoryAction,
} from "@/lib/data/categories-actions";
import type { CategoryRecord } from "@/lib/types";
import { deleteImage } from "@/lib/supabase/storage";

type Props = {
    categories: CategoryRecord[];
};

const initialState: CategoryActionState = {};

export const CategoriesBoard = ({ categories }: Props) => {
    const [search, setSearch] = useState("");
    const [editingId, setEditingId] = useState<string | null>(null);
    const [deleteId, setDeleteId] = useState<string | null>(null);
    const [showCreateForm, setShowCreateForm] = useState(false);
    const [isExpanded, setIsExpanded] = useState(false);

    const filtered = categories.filter(
        (cat) =>
            !search ||
            cat.category_name.toLowerCase().includes(search.toLowerCase()) ||
            cat.description?.toLowerCase().includes(search.toLowerCase()),
    );
```

```
    const handleDelete = async () => {
        if (deleteId) {
            await deleteCategoryAction(deleteId);
            setDeleteId(null);
        }
    };

    return (
        <div className="rounded-2xl border border-white/10 bg-yellow-950/40
p-6">
            <div className="mb-4 flex items-center justify-between">
                <div className="flex-1">
                    <h2 className="text-xl font-semibold text-
white">Categories</h2>
                    <p className="mt-1 text-sm text-yellow-400">
                        Manage menu categories and organize your products
                        {!isExpanded && ` â ¢ ${categories.length} categories`}
                    </p>
                </div>
                <button
                    type="button"
                    onClick={() => setIsExpanded(!isExpanded)}
                    className="flex items-center gap-2 rounded-lg bg-yellow-800
px-4 py-2 text-sm text-white hover:bg-yellow-700"
                >
                    {isExpanded ? (
                        <>
                            <HiChevronUp className="h-4 w-4" />
                            Hide
                        </>
                    ) : (
                        <>
                            <HiChevronDown className="h-4 w-4" />
                            Show
                        </>
                    )}
                </button>
            </div>

            {isExpanded && (
                <div className="space-y-4">
                    <div className="flex items-center justify-between gap-3">
                        <input
                            type="search"
                            placeholder="Search categories..."
                            value={search}
                            onChange={(e) => setSearch(e.target.value)}
                            className="w-full max-w-md rounded-full border
border-white/10 bg-yellow-900/60 px-4 py-3 text-white outline-none focus:border-
indigo-400 focus:ring-2 focus:ring-indigo-400/40"
                        />
                        <button
                            type="button"
                            onClick={() => setShowCreateForm(true)}
                            className="flex items-center gap-2 rounded-full bg-
indigo-600 px-4 py-3 text-sm font-medium text-white hover:bg-indigo-700"
                        >
                            <HiOutlinePlus className="h-5 w-5" />
                            <span className="whitespace-nowrap">Add
Category</span>
                        </button>
                    </div>
```

```
                    {showCreateForm && (
                        <CreateCategoryForm onCancel={() =>
setShowCreateForm(false)} />
                    )}

                    <div className="grid gap-4 md:grid-cols-2 lg:grid-cols-3">
                        {filtered.map((category) => (
                            <CategoryCard
                                key={category.id}
                                category={category}
                                isEditing={editingId === category.id}
                                onEdit={() => setEditingId(category.id)}
                                onCancelEdit={() => setEditingId(null)}
                                onDelete={() => setDeleteId(category.id)}
                            />
                        ))}
                    </div>

                    <DeleteConfirmDialog
                        isOpen={deleteId !== null}
                        onClose={() => setDeleteId(null)}
                        onConfirm={handleDelete}
                        title="Delete Category"
                        message="Are you sure you want to delete this category?
This will fail if any products are using this category."
                    />
                </div>
            )}
        </div>
    );
};

type CreateCategoryFormProps = {
    onCancel: () => void;
};

const CreateCategoryForm = ({ onCancel }: CreateCategoryFormProps) => {
    const [state, formAction] = useFormState(createCategoryAction,
initialState);
    const [uploadedPath, setUploadedPath] = useState<string>("");

    const handleCancel = async () => {
        if (uploadedPath) {
            try {
                await deleteImage(uploadedPath);
            } catch (error) {
                console.error("Failed to delete uploaded image:", error);
            }
        }
        onCancel();
    };

    return (
        <div className="rounded-2xl border border-indigo-500/30 bg-indigo-500/5
p-4">
            <h3 className="mb-3 text-sm font-semibold text-white">Create New
Category</h3>
            <form action={formAction} className="space-y-3">
                <label className="block space-y-2 text-sm">
                    <span className="text-yellow-300">Category Name *</span>
                    <input
                        name="category_name"
                        required
                        className="w-full rounded-lg border border-white/10 bg-
```

```
yellow-900/60 px-3 py-2 text-white outline-none focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
                    />
                </label>

                <label className="block space-y-2 text-sm">
                    <span className="text-yellow-300">Category Image</span>
                    <ImageUpload
                        onChange={(url, path) => setUploadedPath(path || "")}
                        folder="categories"
                        fieldName="image"
                    />
                </label>

                <label className="block space-y-2 text-sm">
                    <span className="text-yellow-300">Description</span>
                    <textarea
                        name="description"
                        rows={3}
                        className="w-full rounded-lg border border-white/10 bg-
yellow-900/60 px-3 py-2 text-white outline-none focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
                    />
                </label>

                <label className="flex items-center gap-2 text-sm">
                    <input
                        name="is_hidden"
                        type="checkbox"
                        className="h-4 w-4 rounded border-white/10 bg-
yellow-900/60 text-indigo-600 focus:ring-2 focus:ring-indigo-400/40"
                    />
                    <span className="text-yellow-300">Hide this category from
public view</span>
                </label>

                {state.error && <p className="text-sm text-
rose-400">{state.error}</p>}
                {state.success && (
                    <p className="text-sm text-emerald-400">{state.success}</p>
                )}

                <div className="flex gap-2">
                    <SubmitButton
                        label="Create Category"
                        loadingLabel="Creating..."
                        className="bg-indigo-600 px-4 py-2 text-sm"
                    />
                    <button
                        type="button"
                        onClick={handleCancel}
                        className="rounded-lg bg-yellow-800 px-4 py-2 text-sm
text-white hover:bg-yellow-700"
                    >
                        Cancel
                    </button>
                </div>
            </form>
        </div>
    );
};

type CategoryCardProps = {
    category: CategoryRecord;
```

```
    isEditing: boolean;
    onEdit: () => void;
    onCancelEdit: () => void;
    onDelete: () => void;
};

const CategoryCard = ({
    category,
    isEditing,
    onEdit,
    onCancelEdit,
    onDelete,
}: CategoryCardProps) => {
    const [state, formAction] = useFormState(updateCategoryAction,
initialState);
    const [uploadedPath, setUploadedPath] = useState<string>("");

    const handleCancel = async () => {
        if (uploadedPath) {
            try {
                await deleteImage(uploadedPath);
            } catch (error) {
                console.error("Failed to delete uploaded image:", error);
            }
        }
        onCancelEdit();
    };

    return (
        <div className="rounded-2xl border border-white/10 bg-yellow-900/40
p-4">
            {isEditing ? (
                <form action={formAction} className="space-y-3">
                    <input type="hidden" name="id" value={category.id} />

                    <label className="block space-y-2 text-sm">
                        <span className="text-yellow-300">Category Name</span>
                        <input
                            name="category_name"
                            defaultValue={category.category_name}
                            required
                            className="w-full rounded-lg border border-white/10
bg-yellow-900/60 px-3 py-2 text-white outline-none focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
                        />
                    </label>

                    <label className="block space-y-2 text-sm">
                        <span className="text-yellow-300">Category Image</span>
                        <ImageUpload
                            defaultValue={category.image}
                            onChange={(url, path) => setUploadedPath(path ||
"")}
                            folder="categories"
                            fieldName="image"
                        />
                    </label>

                    <label className="block space-y-2 text-sm">
                        <span className="text-yellow-300">Description</span>
                        <textarea
                            name="description"
                            rows={3}
                            defaultValue={category.description || ""}
```

```
                            className="w-full rounded-lg border border-white/10
bg-yellow-900/60 px-3 py-2 text-white outline-none focus:border-indigo-400
focus:ring-2 focus:ring-indigo-400/40"
                        />
                    </label>

                    <label className="flex items-center gap-2 text-sm">
                        <input
                            name="is_hidden"
                            type="checkbox"
                            defaultChecked={category.is_hidden ?? false}
                            className="h-4 w-4 rounded border-white/10 bg-
yellow-900/60 text-indigo-600 focus:ring-2 focus:ring-indigo-400/40"
                        />
                        <span className="text-yellow-300">Hide this category
from public view</span>
                    </label>

                    {state.error && <p className="text-sm text-
rose-400">{state.error}</p>}
                    {state.success && (
                        <p className="text-sm text-emerald-400">{state.success}
</p>
                    )}

                    <div className="flex gap-2">
                        <SubmitButton
                            label="Save"
                            loadingLabel="Saving..."
                            className="bg-indigo-600 px-4 py-2 text-sm"
                        />
                        <button
                            type="button"
                            onClick={handleCancel}
                            className="rounded-lg bg-yellow-800 px-4 py-2 text-
sm text-white hover:bg-yellow-700"
                        >
                            Cancel
                        </button>
                    </div>
                </form>
            ) : (
                <>
                    {category.image && (
                        <div className="relative mb-3 aspect-[4/3] w-full
overflow-hidden rounded-lg bg-black/20">
                            <Image
                                src={category.image}
                                alt={category.category_name}
                                fill
                                className="rounded-lg object-contain transition-
transform duration-300 hover:scale-110"
                            />
                        </div>
                    )}

                    <div className="flex items-start justify-between">
                        <div className="flex-1">
                            <h3 className="text-lg font-semibold text-white flex
items-center gap-2">
                                {category.category_name}
                                {category.is_hidden && (
                                    <span className="rounded-full border border-
yellow-500/40 bg-yellow-500/10 px-2 py-0.5 text-[10px] uppercase tracking-wider
```

```
text-yellow-500">
                                        Hidden
                                    </span>
                                )}
                            </h3>
                            {category.description && (
                                <p className="mt-1 text-sm text-yellow-400">
                                    {category.description}
                                </p>
                            )}
                        </div>
                        <div className="flex gap-2">
                            <button
                                type="button"
                                onClick={onEdit}
                                className="rounded-lg bg-yellow-800 px-3 py-1
text-xs text-white hover:bg-yellow-700"
                            >
                                Edit
                            </button>
                            <button
                                type="button"
                                onClick={onDelete}
                                className="rounded-lg bg-rose-600 px-3 py-1
text-xs text-white hover:bg-rose-700"
                            >
                                Delete
                            </button>
                        </div>
                    </div>
                </>
            )}
        </div>
    );
};
</file>

<file path="admin/src/lib/auth/actions.ts">
"use server";

import { redirect } from "next/navigation";
import { createSupabaseServerClient } from "@/lib/supabase/server";
import { fetchAdminProfileWithClient, fetchProfile } from "./session";
import type { AuthActionState } from "./types";
import type { Database } from "@/lib/database.types";

const sanitize = (value: FormDataEntryValue | null) =>
  String(value ?? "").trim();

export async function loginAction(
  _prevState: AuthActionState,
  formData: FormData,
): Promise<AuthActionState> {
  const email = sanitize(formData.get("email")).toLowerCase();
  const password = sanitize(formData.get("password"));
  const redirectTo = sanitize(formData.get("redirectTo")) || "/dashboard";

  if (!email || !password) {
    return { error: "Email and password are required." };
  }

  const supabase = await createSupabaseServerClient();
  const { error } = await supabase.auth.signInWithPassword({
    email,
```

```
    password,
  });

  if (error) {
    return { error: error.message };
  }

  const profile = await fetchProfile(supabase);

  if (!profile) {
    await supabase.auth.signOut();
    return { error: "You do not have admin access." };
  }

  redirect(redirectTo || "/dashboard");
}

export async function registerAction(
  _prevState: AuthActionState,
  formData: FormData,
): Promise<AuthActionState> {
  const email = sanitize(formData.get("email")).toLowerCase();
  const password = sanitize(formData.get("password"));
  const fullName = sanitize(formData.get("fullName")) || undefined;
  const inviteCode = sanitize(formData.get("inviteCode"));

  if (!email || !password) {
    return { error: "Email and password are required." };
  }

  const requiredInvite = process.env.ADMIN_INVITE_CODE;
  if (requiredInvite && inviteCode !== requiredInvite) {
    return { error: "Invalid invite code." };
  }

  const supabase = await createSupabaseServerClient();
  const { data, error } = await supabase.auth.signUp({
    email,
    password,
    options: {
      data: {
        role: "admin",
        display_name: fullName,
      },
    },
  });

  if (error) {
    return { error: error.message };
  }

  if (data.user) {
    const profileData: Database["public"]["Tables"]["profiles"]["Insert"] = {
      id: data.user.id,
      email,
      display_name: fullName,
      role: "admin",
    };
    await supabase.from("profiles").upsert(profileData as never);
  }

  redirect("/login?registered=1");
}
```

```
export async function logoutAction() {
  const supabase = await createSupabaseServerClient();
  await supabase.auth.signOut();
  redirect("/login");
}
</file>

<file path="admin/src/lib/auth/session.ts">
import type { User } from "@supabase/supabase-js";
import { redirect } from "next/navigation";
import { cache } from "react";
import type { SupabaseServerClient } from "@/lib/supabase/server";
import { createSupabaseServerClient } from "@/lib/supabase/server";
import type { AdminProfileSummary } from "@/lib/types";

const PROFILE_COLUMNS = "id,email,display_name,role";

export async function fetchProfile(
  client: SupabaseServerClient,
): Promise<AdminProfileSummary | null> {
  const {
    data: { user },
    error: authError,
  } = await client.auth.getUser();

  if (authError || !user) {
    console.error("Failed to load user:", authError);
    return null;
  }

  const { data: profile, error } = await client
    .from("profiles")
    .select(PROFILE_COLUMNS)
    .eq("id", user.id)
    .single();

  if (error) {
    console.error("Failed to load admin profile", error);
    return null;
  }

  console.log("Profile data:", profile);

  if (!profile) {
    console.log("No profile data found");
    return null;
  }

  const userRole = (profile as { role: string | null }).role;
  console.log("User role:", userRole);

  // Case-insensitive check for admin role
  if (!userRole || userRole.toLowerCase() !== "admin") {
    console.log("Role check failed. Expected 'admin', got:", userRole);
    return null;
  }

  return profile as AdminProfileSummary;
}

export const getSupabaseUser = cache(async (): Promise<User | null> => {
  const supabase = await createSupabaseServerClient();
  const {
    data: { user },
```

```
  } = await supabase.auth.getUser();

  return user;
});

export const getAdminProfile = cache(
  async (): Promise<AdminProfileSummary | null> => {
    const supabase = await createSupabaseServerClient();
    return fetchProfile(supabase);
  },
);

export const requireAdminProfile = async () => {
  const profile = await getAdminProfile();

  if (!profile) {
    redirect("/login?error=not_authorized");
  }

  return profile;
};

export const fetchAdminProfileWithClient = (client: SupabaseServerClient) =>
  fetchProfile(client);
</file>

<file path="client/next.config.ts">
import type { NextConfig } from 'next';

const nextConfig: NextConfig = {
  reactCompiler: true,
  images: {
    remotePatterns: [
      {
        protocol: 'https',
        hostname: 'placehold.co',
        port: '',
        pathname: '/**',
      },
      {
        protocol: 'https',
        hostname: 'images.unsplash.com',
        port: '',
        pathname: '/**',
      },
      {
        protocol: 'https',
        hostname: 'picsum.photos',
        port: '',
        pathname: '/**',
      }, {
        protocol: "https",
        hostname: "qiksfxnjdfpncpsmzvko.supabase.co",
      },
    ],
  },
};

module.exports = nextConfig;
</file>

<file path="client/src/app/globals.css">
@import 'tailwindcss';
@import "tw-animate-css";
```

```
@layer base {
      select {
            width: 100%;
            padding-left: 0.75rem;
            padding-right: 0.75rem;
            padding-top: 0.5rem;
            padding-bottom: 0.5rem;
            border: 1px solid #d1d5db;
            border-radius: 0.375rem;
            outline: none;
      }


      h1 {
            filter: drop-shadow(0 0 3px #f0b100);
            @apply text-4xl md:text-5xl font-bold;
      }

      h2 {
            @apply text-2xl md:text-4xl font-bold border-b-2 p-2 border-
yellow-500 mb-4;
            filter: drop-shadow(0 0 3px #f0b100);
      }

      h3 {
            @apply text-xl md:text-2xl font-semibold;
      }

      strong {
            @apply text-yellow-100 font-bold;
      }

      h4 {
            @apply text-lg md:text-xl font-semibold;
            filter: drop-shadow(0 0 3px #f0b100);
      }

      h5 {
            @apply text-base md:text-lg font-semibold;
            filter: drop-shadow(0 0 3px #f0b100);
      }

      h6 {
            @apply text-sm md:text-base font-semibold;
            filter: drop-shadow(0 0 3px #f0b100);
      }

      p {
            @apply text-balance text-base md:text-lg font-normal;
      }

      body,
      html {
            color: white;
      }

      h1 {
            @apply text-4xl md:text-5xl font-bold;
            filter: drop-shadow(0 0 3px #f0b100);
      }

      /* All other heading tags remain white for consistency */
```

```
      h1,
      h2,
      h3,
      h4,
      h5,
      h6 {
            @apply text-white font-bold text-center my-2 text-shadow-sm text-
shadow-black;
      }

      img {
            object-fit: contain;
      }

      label {
            color: white;
            line-height: 1.625;
            font-weight: bold;
      }

      @media (min-width: 768px) {
            label {
                  font-size: 1.125rem;
            }
      }

      @media (min-width: 768px) {
            p {
                  font-size: 1.125rem;
            }
      }

      li {
            color: white;
            line-height: 1.625;
            font-size: 1rem;
      }

      @media (min-width: 768px) {
            li {
                  font-size: 1.125rem;
            }
      }
}
</file>

<file path="client/src/app/menu/[Products]/[product]/components/comments.tsx">
import { getCommentsForProduct } from '@/lib/supabase/comments';
import Button from '@/components/ui/Button';
import { BiCommentAdd } from 'react-icons/bi';
import Section from '@/components/ui/layout/Section';

interface CommentsProps {
      product: ProductType;
}

/**
 * Displays comments for a given product.
 * Fetches comments for the specific product from Supabase.
 * Note: Comment submission is handled via client-side components.
 * @param product The product for which to display comments.
 */
const Comments: React.FC<CommentsProps> = async ({ product }) => {
      // Determine product id/key used in this app (fallbacks for different
```

```
shapes)
      const productKey = (product as any).ProductID || (product as any).id ||
(product as any).Name || (product as any).slug || (product as any).Slug;

      let comments: any[] = [];
      try {
            if (productKey) {
                  comments = (await getCommentsForProduct(productKey)) || [];
            }
      } catch (err) {
            console.error('Failed to load comments', err);
      }

      return (
            <Section tittle='Reviews & Comments'>
                  {/* Existing Comments */}
                  <ul className='space-y-6'>
                        {comments.length > 0 ? (
                              comments.map((comment) => (
                                    <li
                                          key={comment.id}
                                          className='bg-black/20 border border-
white/50 rounded-md p-4'>
                                          <div className='flex items-start
gap-4'>
                                                <div className='flex-shrink-0
w-10 h-10 bg-yellow-600 rounded-full flex items-center justify-center text-white
font-bold text-lg'>
                                                      {(comment.user_name ||
'U').charAt(0).toUpperCase()}
                                                </div>
                                                <div className='flex-1'>
                                                      <div className='flex
justify-between items-center mb-1'>
                                                            <span
className='font-bold text-white'>{comment.user_name || 'Anonymous'}</span>
                                                            <span
className='text-xs text-white/70'>
                                                                  {new
Date(comment.created_at).toLocaleDateString()}
                                                            </span>
                                                      </div>
                                                      <p className='text-
white'>{comment.body}</p>
                                                      <button className='mt-3
px-3 py-1 bg-yellow-600 text-white text-xs font-semibold rounded-md hover:bg-
yellow-600/100 transition-colors duration-200'>
                                                            Reply
                                                      </button>
                                                </div>
                                          </div>
                                    </li>
                              ))
                        ) : (
                              <li className='text-center text-white/70 py-4'>
                                    No comments yet. Be the first to review!
                              </li>
                        )}
                  </ul>
            </Section>
      );
};

export default Comments;
```

```
</file>

<file
path="client/src/app/menu/[Products]/[product]/components/RelatedProducts.tsx">
import React from 'react';
import Image from 'next/image';
import Link from 'next/link';


import Section from '@/components/ui/layout/Section';

interface RelatedProductsProps {
      currentProduct: ProductType;
      categoryName: string;
      products: ProductType[];
}

const RelatedProducts: React.FC<RelatedProductsProps> = ({ currentProduct,
categoryName, products }) => {
      if (products.length === 0) {
            return null;
      }

      return (
            <Section tittle='You Might Also Like'>
                  <div className='grid grid-cols-3 sm:grid-cols-2 md:grid-cols-4
gap-2'>
                        {products.map((product) => (
                              <Link
                                    key={product.ProductID}
                                    href={`/menu/${categoryName}/$
{product.Name.toLowerCase().replace(/\s+/g, '-')}`}
                                    className='group relative bg-yellow-700
rounded-md p-2 pt-6  flex flex-col transition-transform duration-300
hover:scale-105 hover:border-white'>
                                    {product.Image && (
                                          <div className='relative w-full
aspect-square  overflow-hidden rounded-md'>
                                                <Image
                                                      src={product.Image}
                                                      alt={product.Name}
                                                      fill
                                                      className=' transition-
transform duration-300 group-hover:scale-110'
                                                />
                                          </div>
                                    )}
                                    <div className='flex-grow  flex flex-col'>
                                          <h1 className='font-semibold text-
white text-base mb-2 line-clamp-2 flex-grow'>
                                                {product.Name}
                                          </h1>
                                          <p className='text-white font-bold
text-md mt-auto'>R{product.Price.toFixed(2)}</p>
                                          {product.badge && (
                                                <span className='absolute w-full
top-0 right-0 px-2 py-1 rounded-t-md text-xs font-bold bg-yellow-600 text-
white'>
                                                      {product.badge}
                                                </span>
                                          )}
                                    </div>
                              </Link>
                        ))}
```

```
                </div>
            </Section>
        );
};

export default RelatedProducts;
</file>

<file path="client/src/app/menu/[Products]/utils/likesUtils.ts">
import { createClient } from '@/lib/supabase/client';

export interface ProductLikes {
  [productKey: string]: {
    likes: number;
  };
}

export const fetchAllProductLikes = async (productKeys: string[]):
Promise<ProductLikes> => {
  const likesData: ProductLikes = {};
  const supabase = createClient();

  try {
    // Fetch likes data for each product from Supabase
    const promises = productKeys.map(async (productKey) => {
      try {
        const { data, error } = await supabase
          .from('products')
          .select('likes')
          .eq('id', productKey)
          .single();

        if (!error && data) {
          // @ts-ignore - Supabase type inference issue with Database types
          likesData[productKey] = { likes: data.likes || 0 };
        } else {
          likesData[productKey] = { likes: 0 };
        }
      } catch (err) {
        likesData[productKey] = { likes: 0 };
      }
    });

    await Promise.all(promises);
    return likesData;
  } catch (error) {
    console.error('Error fetching product likes:', error);
    // Return default values if there's an error
    productKeys.forEach(productKey => {
      likesData[productKey] = { likes: 0 };
    });
    return likesData;
  }
};
</file>

<file path="client/src/app/menu/cart/page.tsx">
import React, { Suspense } from 'react';
import Loading from '@/components/ui/Loading';
import { IoClose } from 'react-icons/io5';
import CartProductForm from '@/lib/forms/CartProductForm';
import CartFooter from '@/app/menu/cart/components/CartFooter';
import AppLink from '@/components/ui/Link';
import Main from '@/components/ui/layout/Main';
```

```
const CartPage: React.FC = () => {
      return (
            <Main >
                  <AppLink className="mb-2 bg-red-500/10 backdrop-blur-sm border
border-red-500/30 hover:border-red-500/50 transition-all duration-200
hover:shadow-red-500/20 rounded-xl" href="/menu" >
                        <IoClose /> Close Cart
                  </AppLink>
                  <Suspense fallback={<Loading message="Loading Cart..." />}>
                        <CartProductForm />
                  </Suspense>
                  <Suspense fallback={<Loading message="Loading Cart
footer..." />}>
                        <CartFooter />
                  </Suspense>
            </Main>

      );
};

export default CartPage;
</file>

<file path="client/src/app/page.tsx">
import AboutUsSnippet from '@/app/home-components/AboutUsSnippet';
import ContactSection from '@/app/home-components/ContactSection';
import FeaturedItemsServices from '@/app/home-components/FeaturedItemsServices';
import HeroSection from '@/app/home-components/HeroSection';
import KeyDifferentiators from '@/app/home-components/KeyDifferentiators';
import TargetMarketCallout from '@/app/home-components/TargetMarketCallout';
import Testimonials from '@/app/home-components/Testimonials';
import Main from '@/components/ui/layout/Main';
const HomePage = () => {
      return (
            <Main>
                  <HeroSection />
                  <AboutUsSnippet />
                  <KeyDifferentiators />
                  <FeaturedItemsServices />
                  <TargetMarketCallout />
                  <Testimonials />
                  <ContactSection />
            </Main>
      );
};
export default HomePage;
</file>

<file path="client/src/app/photo/page.tsx">
import Main from '@/components/ui/layout/Main';
import { HiCamera } from 'react-icons/hi2';
import Image from 'next/image';
import AppLink from '@/components/ui/Link';

export default function PhotoBoothPage() {
      return (
            <Main
                  tittle='360 Photo Booth'
                  Icon={HiCamera}
                  heading='Capture Your Moments in 360Â°'>
                  {/* Hero Section */}
                  <div className='grid md:grid-cols-2 gap-8 mb-12'>
```

```
                        <div className='flex flex-col justify-center'>
                              <h2 className='text-3xl font-bold text-white
mb-4'>Premium 360Â° Photo Booth Experience</h2>
                              <p className='text-yellow-300 mb-6'>
                                    Step into the spotlight with our state-of-
the-art 360Â° photo booth! Create stunning, shareable content that captures
every angle of your special moments. Perfect for events, celebrations, and
creating unforgettable memories.
                              </p>
                              <AppLink
                                    href='/photo/booking'
                                    className='inline-flex items-center gap-2
px-6 py-3 bg-gradient-to-r from-amber-600 to-yellow-600 hover:from-amber-500
hover:to-yellow-500 rounded-xl shadow-lg hover:shadow-amber-500/50 transition-
all duration-200 font-semibold text-white w-fit'>
                                    <HiCamera size={20} />
                                    <span>Book Your Session</span>
                              </AppLink>
                        </div>
                        <div className='relative h-64 md:h-auto rounded-xl
overflow-hidden border border-yellow-500/30 shadow-xl'>
                              <Image
                                    src='/PhotoBoot.jpg'
                                    alt='360 Photo Booth'
                                    fill
                                    className='object-cover'
                              />
                        </div>
                  </div>

                  {/* Features Grid */}
                  <div className='grid md:grid-cols-3 gap-6 mb-12'>
                        <div className='bg-gradient-to-br from-yellow-900/20 to-
amber-900/20 backdrop-blur-md border border-yellow-500/30 rounded-xl p-6
hover:border-yellow-400/50 transition-all duration-300'>
                              <div className='bg-gradient-to-br from-
amber-500/20 to-yellow-500/20 p-3 rounded-lg border border-amber-400/30 w-fit
mb-4'>
                                    <HiCamera className='text-amber-400
text-3xl' />
                              </div>
                              <h3 className='text-xl font-bold text-white
mb-2'>360Â° Coverage</h3>
                              <p className='text-yellow-300'>
                                    Capture every angle with our rotating camera
system that creates stunning slow-motion videos.
                              </p>
                        </div>

                        <div className='bg-gradient-to-br from-yellow-900/20 to-
amber-900/20 backdrop-blur-md border border-yellow-500/30 rounded-xl p-6
hover:border-yellow-400/50 transition-all duration-300'>
                              <div className='bg-gradient-to-br from-
amber-500/20 to-yellow-500/20 p-3 rounded-lg border border-amber-400/30 w-fit
mb-4'>
                                    <HiCamera className='text-amber-400
text-3xl' />
                              </div>
                              <h3 className='text-xl font-bold text-white
mb-2'>Instant Sharing</h3>
                              <p className='text-yellow-300'>
                                    Get your videos instantly via email or QR
code. Share directly to social media!
                              </p>
```

```
                        </div>

                        <div className='bg-gradient-to-br from-yellow-900/20 to-
amber-900/20 backdrop-blur-md border border-yellow-500/30 rounded-xl p-6
hover:border-yellow-400/50 transition-all duration-300'>
                              <div className='bg-gradient-to-br from-
amber-500/20 to-yellow-500/20 p-3 rounded-lg border border-amber-400/30 w-fit
mb-4'>
                                    <HiCamera className='text-amber-400
text-3xl' />
                              </div>
                              <h3 className='text-xl font-bold text-white
mb-2'>Professional Quality</h3>
                              <p className='text-yellow-300'>
                                    High-definition cameras and professional
lighting ensure stunning results every time.
                              </p>
                        </div>
                  </div>

                  {/* Pricing Info */}
                  <div className='bg-gradient-to-br from-yellow-900/20 to-
amber-900/20 backdrop-blur-md border border-yellow-500/30 rounded-xl p-8'>
                        <h2 className='text-2xl font-bold text-white mb-6 text-
center'>Simple Pricing</h2>
                        <div className='grid md:grid-cols-2 gap-6'>
                              <div className='bg-white/5 rounded-lg p-6 border
border-white/10'>
                                    <h3 className='text-xl font-bold text-
amber-400 mb-2'>Individual Session</h3>
                                    <p className='text-3xl font-bold text-white
mb-4'>R50 <span className='text-lg text-yellow-400'>/ person</span></p>
                                    <ul className='text-yellow-300 space-y-2'>
                                          <li>â€“ 1x 360Â° video</li>
                                          <li>â€“ Instant digital delivery</li>
                                          <li>â€“ Social media ready</li>
                                    </ul>
                              </div>
                              <div className='bg-gradient-to-br from-
amber-600/20 to-yellow-600/20 rounded-lg p-6 border border-amber-400/40'>
                                    <h3 className='text-xl font-bold text-
amber-400 mb-2'>Event Package</h3>
                                    <p className='text-3xl font-bold text-white
mb-4'>Contact <span className='text-lg text-yellow-400'>/ event</span></p>
                                    <ul className='text-yellow-300 space-y-2'>
                                          <li>â€“ Unlimited sessions</li>
                                          <li>â€“ Custom branding options</li>
                                          <li>â€“ Dedicated attendant</li>
                                          <li>â€“ Props included</li>
                                    </ul>
                              </div>
                        </div>
                  </div>
            </Main>
      );
}
</file>

<file path="client/src/components/Navbar/Navbar.tsx">
import { createClient } from '@/lib/supabase/server';
import NavbarClient from './NavbarClient';

const Navbar = async () => {
      const supabase = await createClient();
```

```
	const {
		data: { user },
	} = await supabase.auth.getUser();

	return <NavbarClient user={user} />;
};

export default Navbar;
</file>

<file path="client/src/components/ui/layout/Main.tsx">
interface MainProps {
	Icon?: React.ElementType;
	className?: string;
	heading?: string;
	tittle?: string;
	children: React.ReactNode;
}
export default function Main({ children, Icon, heading, tittle, className }:
MainProps) {
	return (
		<main
			className={
				className
					? className
					: 'flex flex-col items-center justify-start w-full
min-h-screen p-2 md:px-4 '
			}>
			<header>
				<span className='flex items-center justify-center
gap-3'>
					{Icon && <Icon className='text-3xl text-
yellow-500' />}
					{tittle && <h1>{tittle}</h1>}
				</span>
				<p className='text-white/70 text-center mt-2 flex-
grow'>{heading}</p>
			</header>
			{children}
		</main>
	);
}
</file>

<file path="client/src/components/ui/TextInput.tsx">
import React from 'react';
import classNames from 'classnames';

export type TextInputVariant = 'default' | 'error';
export type CustomTextInputSize = 'sm' | 'md' | 'lg'; // Renamed to avoid
conflict

interface TextInputProps extends React.InputHTMLAttributes<HTMLInputElement> {
	variant?: TextInputVariant;
	customSize?: CustomTextInputSize; // Using the new name
	loading?: boolean;
	label?: string;
	fullWidth?: boolean;
	icon?: React.ReactNode;
}

const baseClasses = 'block rounded-md transition-all duration-200 focus:outline-
none focus:ring-2 ';
```

```
const variantClasses = {
      default: 'border-white/30 bg-yellow-900/60 text-white focus:border-
amber-400 focus:ring-amber-400/40',
      error: 'border-red-500/50 bg-yellow-900/60 text-white focus:border-red-400
focus:ring-red-400/40',
};

// Use the new custom size type here
const sizeClasses = {
      sm: 'px-3 py-1 text-sm',
      md: 'px-4 py-2 text-base',
      lg: 'px-5 py-2.5 text-lg',
};

const TextInput: React.FC<TextInputProps> = ({
      label,
      variant = 'default',
      customSize = 'md',
      loading = false,
      disabled,
      className,
      fullWidth = true,
      icon,
      ...props
}) => {
      const inputId = props.id || `text-input-$
{Math.random().toString(36).substring(2, 9)}`;

      const containerClasses = classNames('relative', { 'w-full': fullWidth });

      const inputClasses = classNames(
            baseClasses,
            variantClasses[variant],
            sizeClasses[customSize],
            className,
            {
                  'opacity-50 cursor-not-allowed': disabled || loading,
                  'w-full': fullWidth,
                  'pl-10': icon,
            }
      );

      return (
            <div className={containerClasses}>
                  {label && (
                        <label
                              htmlFor={inputId}
                              className='block text-sm font-medium text-white
mb-1'>
                              {label}
                        </label>
                  )}
                  {icon && (
                        <div className='absolute inset-y-0 left-0 pl-3 flex
items-center pointer-events-none'>
                              {icon}
                        </div>
                  )}
                  <input
                        id={inputId}
                        className={inputClasses}
                        disabled={disabled || loading}
                        {...props}
                  />
```

```
            </div>
      );
};

export default TextInput;
</file>

<file path="client/src/lib/forms/CartProductForm.tsx">
'use client';

import ProductDetails from '@/app/menu/components/ProductDetails';
import Button from '@/components/ui/Button';
import Section from '@/components/ui/layout/Section';
import { useCart } from '@/lib/context/CartContext';
import { HiMinusSm, HiPlusSm, HiTrash, HiShoppingCart } from 'react-icons/hi';

function CartProductForm() {
      const { cartItems, removeFromCart, minisItemQuantity, addItem } =
useCart();

      if (!cartItems || cartItems.length === 0) {
            return (
                  <div className='text-center bg-gradient-to-br from-
yellow-900/90 to-yellow-800/90 backdrop-blur-xl p-12 rounded-2xl border border-
white/10 shadow-2xl'>
                        <div className='flex flex-col items-center gap-4'>
                              <div className='bg-gradient-to-br from-
amber-500/20 to-yellow-500/20 p-6 rounded-full border border-amber-400/30'>
                                    <HiShoppingCart className='text-6xl text-
amber-400' />
                              </div>
                              <h2 className='text-3xl font-bold text-white'>
                                    Your Cart is Empty
                              </h2>
                              <p className='text-yellow-400 max-w-md'>
                                    Start adding delicious items to your cart
and place your order!
                              </p>
                        </div>
                  </div>
            );
      }

      return (
            <Section
                  tittle='Your Items'
                  className='p-4 max-h-[70vh] overflow-y-auto custom-scrollbar'>
                  <ul className='space-y-3 px-2'>
                        {cartItems.map((item) => (
                              <li
                                    className=' group rounded-xl border border-
white/10 bg-gradient-to-br from-yellow-900/80 to-yellow-800/80 backdrop-blur-sm
hover:border-amber-400/30 transition-all duration-300 hover:shadow-lg
hover:shadow-amber-500/10'
                                    key={item.ProductID}>
                                    <article className='p-4 flex flex-col
md:flex-row items-center justify-between gap-4'>
                                          <ProductDetails item={item} />

                                          <div className='flex items-center
gap-3 w-full md:w-auto'>
                                                {/* Quantity Controls with modern
design */}
                                                <div className='flex items-center
```

```
gap-2 bg-white/5 backdrop-blur-sm px-2 py-1.5 rounded-xl border
border-white/10'>
                                                      <Button
                                                            title='Decrease
Quantity'
                                                            onClick={() =>
minisItemQuantity(item)}
                                                            className='p-2 text-
white rounded-lg bg-gradient-to-br from-amber-600/80 to-yellow-600/80
hover:from-amber-500 hover:to-yellow-500 transition-all duration-200
disabled:opacity-50 border-0 shadow-md hover:shadow-amber-500/50'>
                                                            <HiMinusSm
size={18} />
                                                      </Button>

                                                      <span className='px-4 text-
xl font-bold text-white min-w-[3rem] text-center'>
                                                            {item.quantity}
                                                      </span>

                                                      <Button
                                                            title='Increase
Quantity'
                                                            onClick={() =>
addItem(item)}
                                                            className='p-2 text-
white rounded-lg bg-gradient-to-br from-amber-600/80 to-yellow-600/80
hover:from-amber-500 hover:to-yellow-500 transition-all duration-200
disabled:opacity-50 border-0 shadow-md hover:shadow-amber-500/50'>
                                                            <HiPlusSm
size={18} />
                                                      </Button>
                                                </div>

                                                {/* Remove Button with improved
styling */}
                                                <Button
                                                      title='Remove Item'
                                                      onClick={() =>
removeFromCart(item.ProductID)}
                                                      className='flex items-
center justify-center gap-2 px-4 py-2.5 text-red-300 rounded-xl bg-red-500/10
hover:bg-red-500/20 border border-red-500/30 hover:border-red-500/50 transition-
all duration-200 font-semibold shadow-md hover:shadow-red-500/20'>
                                                      <HiTrash size={18} />
                                                      <span className='hidden
sm:inline'>Remove</span>
                                                </Button>
                                          </div>
                                    </article>
                              </li>
                        ))}
                  </ul>
            </Section>
      );
}

export default CartProductForm;
</file>

<file path="client/src/lib/forms/CommentsForm.tsx">
import { getCommentsForProduct } from '@/lib/supabase/comments';
import { redirect } from 'next/navigation';
import Section from '@/components/ui/layout/Section';
```

```
import Button from '@/components/ui/Button';

interface CommentsProps {
      ProductName: string;
}

const CommentsForm: React.FC<CommentsProps> = async ({ ProductName }) => {
       // NOTE: GetUser should be called server-side via Supabase session;
placeholder for now
       // You should pass user as a prop or fetch from Supabase session
      const user: any = null; // TODO: Get authenticated user from Supabase
session

       if (!user) {
             return redirect('/login');
       }

       let comments: Array<{ id: string; user_name: string | null; created_at:
string; body: string }> = [];
       try {
             comments = (await getCommentsForProduct(ProductName)) || [];
       } catch (err) {
             console.error('Failed to load comments', err);
       }

       return (
             <Section>
                   <h2>Comments</h2>
                   <ul className='space-y-4'>
                         {comments.map((comment) => (
                               <li
                                     key={comment.id}
                                     className='border-b border-white/20 pb-4'>
                                     <div className='flex items-start space-
x-3'>
                                           <div className='flex-shrink-0 w-8 h-8
bg-yellow-500 rounded-full flex items-center justify-center text-white font-
bold'>
                                                 {(comment.user_name ||
'U').charAt(0).toUpperCase()}
                                           </div>
                                           <div className='flex-1'>
                                                 <div className='flex justify-
between items-center'>
                                                       <span className='font-
semibold '>{comment.user_name || 'Anonymous'}</span>
                                                       <span className='text-xs
'>{new Date(comment.created_at).toLocaleString()}</span>
                                                 </div>
                                                 <p
className='mt-1'>{comment.body}</p>
                                                 <button className='mt-2 text-sm
text-yellow-400 hover:underline'>REPLY</button>
                                           </div>
                                     </div>
                               </li>
                         ))}
                   </ul>
                   <div className='mt-6 flex space-x-3'>
                         <div className='flex-shrink-0 w-8 h-8 bg-yellow-500
rounded-full flex items-center justify-center text-white font-bold'>
                               {(user?.user_metadata?.displayName || user?.email
|| 'U').charAt(0).toUpperCase()}
                         </div>
```

```
                        <div className='flex-1'>
                              <textarea
                                    className='w-full p-3 border
border-white/50 rounded-md bg-black/20 text-white placeholder-white/60
focus:outline-none focus:ring-2 focus:ring-yellow-500 focus:border-yellow-500
transition-all duration-200'
                                    placeholder={`What are your thoughts on $
{ProductName}?`}
                                    rows={2}></textarea>
                              <Button>Send</Button>
                        </div>
                  </div>
            </Section>
      );
};

export default CommentsForm;
</file>

<file path="client/src/lib/forms/UserAddressForm.tsx">
'use client';

import React, { useState, useEffect } from 'react';
import { useActionState } from 'react';
import TextInput from '@/components/ui/TextInput';
import Button from '@/components/ui/Button';
import { FaMapMarkerAlt, FaSave } from 'react-icons/fa';
import { TbAccessPoint } from 'react-icons/tb';
import Loading from '@/components/ui/Loading';
import { useAuth } from '@/lib/supabase/auth/useAuth';
import { updateAddressAction } from '@/lib/actions/profile';

const UserAddressForm: React.FC = () => {
      const { user, loading } = useAuth();
      const [showAddressInput, setShowAddressInput] = useState(false);
      const [state, formAction, isPending] = useActionState(updateAddressAction,
null);

      const [addressData, setAddressData] = useState({
            address: '',
            city: '',
            state: '',
            zipCode: '',
            country: '',
      });

      const [errors, setErrors] = useState({
            address: false,
            city: false,
            state: false,
            zipCode: false,
            country: false,
      });

      useEffect(() => {
            if (user) {
                  setAddressData({
                        address: user.user_metadata?.address || '',
                        city: user.user_metadata?.city || '',
                        state: user.user_metadata?.state || '',
                        zipCode: user.user_metadata?.zipCode || '',
                        country: user.user_metadata?.country || '',
                  });
            }
```

```
      }, [user]);

      // Close form on successful save
      useEffect(() => {
            if (state?.success) {
                  setShowAddressInput(false);
            }
      }, [state?.success]);

      const validateFields = () => {
            const newErrors = {
                  address: !addressData.address.trim(),
                  city: !addressData.city.trim(),
                  state: !addressData.state.trim(),
                  zipCode: !addressData.zipCode.trim(),
                  country: !addressData.country.trim(),
            };
            setErrors(newErrors);
            return !Object.values(newErrors).some(Boolean);
      };

      const handleSubmit = (e: React.FormEvent<HTMLFormElement>) => {
            if (!validateFields() || !user?.id) {
                  e.preventDefault();
                  return;
            }
      };

      const handleInputChange = (e: React.ChangeEvent<HTMLInputElement>) => {
            const { name, value } = e.target;
            setAddressData((prev) => ({ ...prev, [name]: value }));
            setErrors((prev) => ({ ...prev, [name]: false }));
      };

      if (loading) return <Loading />;
      if (!user) return null;

      const fullAddress = `${user.user_metadata?.address || ''}, $
{user.user_metadata?.city || ''}, ${user.user_metadata?.state || ''}, $
{user.user_metadata?.zipCode || ''}, ${user.user_metadata?.country || ''}`;

      return (
            <article>
                  <h3 className='flex items-center gap-2 text-xl font-bold
mb-3'>
                        <FaMapMarkerAlt /> Shipping Address
                  </h3>

                  {user.user_metadata?.address && !showAddressInput ? <p
className='text-white'>{fullAddress}</p> : null}

                  {showAddressInput ? (
                        <div className='mt-4 flex flex-col gap-3'>
                              {state?.error && (
                                    <div className='text-yellow-500 mb-2 bg-
yellow-900/20 p-2 rounded-md text-sm'>{state.error}</div>
                              )}
                              {state?.success && (
                                    <div className='text-white mb-2 bg-
green-900/20 p-2 rounded-md text-sm'>Address saved successfully!</div>
                              )}
                              <form action={formAction} onSubmit={handleSubmit}
className='flex flex-col gap-3'>
                                    <TextInput
```

```
                                        type='text'
                                        name='address'
                                        placeholder='e.g., 123 Main St, Apt
4B'
                                        label='Street Address'
                                        value={addressData.address}
                                        onChange={handleInputChange}
                                        className={errors.address ? 'border-
yellow-500 ring-yellow-500' : ''}
                                        required
                                    />
                                    {errors.address && <p className='text-white
text-sm mt-1'>Street address is required.</p>}

                                    <TextInput
                                        type='text'
                                        name='city'
                                        placeholder='e.g., Phalaborwa'
                                        label='City'
                                        value={addressData.city}
                                        onChange={handleInputChange}
                                        className={errors.city ? 'border-
yellow-500 ring-yellow-500' : ''}
                                        required
                                    />
                                    {errors.city && <p className='text-white
text-sm mt-1'>City is required.</p>}

                                    <TextInput
                                        type='text'
                                        name='state'
                                        placeholder='e.g., Limpopo'
                                        label='State'
                                        value={addressData.state}
                                        onChange={handleInputChange}
                                        className={errors.state ? 'border-
yellow-500 ring-yellow-500' : ''}
                                        required
                                    />
                                    {errors.state && <p className='text-white
text-sm mt-1'>State is required.</p>}

                                    <div className='flex gap-2'>
                                        <TextInput
                                            type='text'
                                            name='zipCode'
                                            placeholder='e.g., 1390'
                                            label='Zip Code'
                                            value={addressData.zipCode}
                                            onChange={handleInputChange}
                                            className={`flex-grow $
{errors.zipCode ? 'border-yellow-500 ring-yellow-500' : ''}`}
                                            required
                                        />
                                        <TextInput
                                            type='text'
                                            name='country'
                                            placeholder='e.g., South Africa'
                                            label='Country'
                                            value={addressData.country}
                                            onChange={handleInputChange}
                                            className={`flex-grow $
{errors.country ? 'border-yellow-500 ring-yellow-500' : ''}`}
                                            required
```

```
                                    />
                                </div>
                                {(errors.zipCode || errors.country) && (
                                    <p className='text-white text-sm
mt-1'>
                                        {errors.zipCode && 'Zip code is
required.'}
                                        {errors.zipCode && errors.country
&& ' '}
                                        {errors.country && 'Country is
required.'}
                                    </p>
                                )}

                                <Button
                                    type='submit'
                                    loading={isPending}
                                    disabled={isPending}
                                    className='h-10 px-4 py-2 inline-flex
items-center justify-center gap-2 bg-yellow-600 text-white rounded-md hover:bg-
yellow-700 focus:outline-none focus:ring-2 focus:ring-white transition
duration-200 mt-2'>
                                    <FaSave /> Save Address
                                </Button>
                            </form>
                        </div>
                    ) : (
                        <button
                            onClick={() => setShowAddressInput(true)}
                            className='inline-flex items-center gap-2 px-4
py-2 bg-black/50 text-white rounded-md hover:bg-yellow-600 focus:outline-none
focus:ring-2 focus:ring-yellow-500 focus:ring-opacity-50 transition
duration-200'>
                            <TbAccessPoint /> {user.user_metadata?.address ?
'Edit Address' : 'Add Address'}
                        </button>
                    )}
                </article>
            );
};

export default UserAddressForm;
</file>

<file path="client/src/lib/supabase/likes.ts">
import { createClient } from './client';

export const getLikeCount = async (table: string, id: string) => {
  const supabaseBrowser = createClient();
  const { data, error } = await supabaseBrowser
    .from(table)
    .select('likes')
    .eq('id', id)
    .single();
  if (error) throw error;
  return (data as any)?.likes?.length ?? 0;
};

export const toggleLike = async (table: string, id: string, userId: string) => {
  const supabaseBrowser = createClient();

  // Fetch current likes
  const { data: current, error: fetchErr } = await supabaseBrowser
    .from(table)
```

```
    .select('likes')
    .eq('id', id)
    .single();

  if (fetchErr) throw fetchErr;

  const currentLikes = (current as any)?.likes ?? [];
  const hasLiked = currentLikes.includes(userId);

  let newLikes;
  if (hasLiked) {
    newLikes = currentLikes.filter((uid: string) => uid !== userId);
  } else {
    newLikes = [...currentLikes, userId];
  }

  const { data, error } = await supabaseBrowser
    .from(table)
    .update({ likes: newLikes } as never)
    .eq('id', id)
    .select()
    .single();

  if (error) throw error;
  return data;
};
</file>

<file path="client/src/lib/supabase/server.ts">
import { createServerClient } from '@supabase/ssr';
import { cookies } from 'next/headers';
import { Database } from '@/lib/types/database.types';

import { SupabaseClient } from '@supabase/supabase-js';

export async function createClient(): Promise<SupabaseClient<Database>> {
  const cookieStore = await cookies();

  return createServerClient<Database>(
    process.env.NEXT_PUBLIC_SUPABASE_URL!,
    process.env.NEXT_PUBLIC_SUPABASE_PUBLISHABLE_KEY!,
    {
      cookies: {
        getAll() {
          return cookieStore.getAll();
        },
        setAll(cookiesToSet) {
          try {
            cookiesToSet.forEach(({ name, value }) => cookieStore.set(name,
value));
          } catch {
            // The `setAll` method was called from a Server Component.
            // This can be ignored if you have middleware refreshing
            // user sessions.
          }
        },
      },
    }
  );
}
</file>

<file path="admin/src/app/(dashboard)/menu/page.tsx">
import { AddProductPanel } from "@/components/menu/products/AddProductPanel";
```

```
import { ProductBoard } from "@/components/menu/products/ProductBoard";
import { CategoriesBoard } from "@/components/menu/CategoriesBoard";
import { fetchProductCatalog } from "@/lib/data/products";
import { fetchCategories } from "@/lib/data/categories";
import { fetchComments } from "@/lib/data/comments";
import { fetchUserFavorites } from "@/lib/data/favorites";

export default async function MenuPage() {
  const [products, categories, comments, favorites] = await Promise.all([
    fetchProductCatalog(),
    fetchCategories(),
    fetchComments(),
    fetchUserFavorites(),
  ]);

  const categoryNames = categories.map((category) => category.category_name);

  return (
    <div className="space-y-8">
      {/* Categories Section */}
      <CategoriesBoard categories={categories} />

      {/* Products Section */}
      <div className="grid gap-6 lg:grid-cols-[2fr,1fr]">
        <ProductBoard
          products={products}
          comments={comments}
          favorites={favorites}
        />
        <AddProductPanel categories={categoryNames} />
      </div>
    </div>
  );
}
</file>

<file path="client/src/app/layout.tsx">
import type { Metadata } from 'next';
import './globals.css';
import './layout.css';

import Navbar from '@/components/Navbar/Navbar';
import { Analytics } from '@vercel/analytics/next';

export default function RootLayout({
      children,
}: Readonly<{
      children: React.ReactNode;
}>) {
      return (
            <html lang='en'>
                  <body className={`antialiased min-h-screen`}>
                        <Navbar />
                        {children}
                        <Analytics />
                  </body>
            </html>
      );
}
</file>

<file path="client/src/app/menu/[Products]/[product]/page.tsx">
import Image from 'next/image';
import SocialButtons from '../components/SocialButtons';
```

```
import Comments from './components/comments';
import ProductInfo from './components/ProductInfo';
import RelatedProducts from './components/RelatedProducts';
import FavoriteButton from './components/FavoriteButton';
import AddtoCart from '@/app/menu/[Products]/[product]/components/AddtoCart';
import Main from '@/components/ui/layout/Main';
import Section from '@/components/ui/layout/Section';
import { createClient } from '@/lib/supabase/server';

async function ProductPage({ params }: { params: Promise<{ product: string;
Products: string }> }) {
      const { product: productId, Products: categorySlug } = await params;
      const supabase = await createClient();

      // Reconstruct name from slug - handle both hyphens and spaces
      // The slug might be "coca-cola" but the DB has "Coca-Cola" or "Coca Cola"
      // We'll try multiple patterns to match both hyphens and spaces
      const productNameSlug = productId.replace(/-/g, ' ');
      const categoryName = categorySlug.replace(/-/g, ' ');

      // Fetch product
      type ProductRow = {
            id: string;
            name: string;
            description: string | null;
            price: number | null;
            image_url: string | null;
            badge: string | null;
            likes: number | null;
      };

      // Try multiple search patterns to handle hyphens in product names
      // PostgreSQL ilike uses SQL LIKE patterns, so we use % wildcards
      let productData = null;
      let error = null;

      // Strategy: Split slug into words and create flexible pattern
      // "coca-cola" -> ["coca", "cola"] -> pattern "%coca%cola%"
      // This matches "Coca-Cola", "Coca Cola", "coca cola", etc.
      const words = productId.split('-').filter(w => w.length > 0);
      const flexiblePattern = words.length > 1
            ? `%${words.join('%')}%`  // "%coca%cola%" matches both hyphens and
spaces
            : `%${productId}%`;        // Single word, just use wildcards

      // Try flexible pattern first (handles hyphens, spaces, and case)
      const { data: patternMatch, error: patternError } = await supabase
            .from('products')
            .select('*')
            .ilike('name', flexiblePattern)
            .limit(10); // Get multiple matches to find the best one

      if (patternMatch && patternMatch.length > 0 && !patternError) {
            // If multiple matches, prefer exact match (with spaces or hyphens)
            const typedMatches = patternMatch as ProductRow[];
            const exactWithSpaces = typedMatches.find(p =>
                  p.name.toLowerCase().replace(/[-\s]/g, ' ') ===
productNameSlug.toLowerCase()
            );
            const exactWithHyphens = typedMatches.find(p =>
                  p.name.toLowerCase().replace(/[-\s]/g, '-') ===
productId.toLowerCase()
            );
```

```
            productData = exactWithSpaces || exactWithHyphens ||
typedMatches[0];
      } else {
            // Fallback: try exact match with spaces
            const { data: exactMatch, error: exactError } = await supabase
                  .from('products')
                  .select('*')
                  .ilike('name', productNameSlug)
            .single();

            if (exactMatch && !exactError) {
                  productData = exactMatch as ProductRow;
            } else {
                  error = exactError || patternError;
            }
      }

      if (error || !productData) {
            return <div>Product not found</div>;
      }

      const product = productData as ProductRow;

      // Map to UI ProductType
      const productUI: ProductType = {
            ProductID: product.id,
            Name: product.name,
            Description: [product.description || ''],
            Price: product.price || 0,
            Image: product.image_url || '/Menus/placeholder.png',
            badge: product.badge || undefined,
            rating: product.likes ? 5 : undefined,
      };

      // Fetch related products (same category, exclude current)
      const { data: relatedData } = await supabase
            .from('products')
            .select('*')
            .ilike('category', categoryName)
            .neq('id', productUI.ProductID)
            .limit(4);

      const relatedProducts: ProductType[] = ((relatedData as ProductRow[]) ||
[]).map((p) => ({
            ProductID: p.id,
            Name: p.name,
            Description: [p.description || ''],
            Price: p.price || 0,
            Image: p.image_url || '/Menus/placeholder.png',
            badge: p.badge || undefined,
            rating: p.likes ? 5 : undefined,
      }));

      return (
            <Main>
                  {/* Main Product Section */}
                  <Section>
                        {/* Product Image */}
                        {productUI.Image && (
                              <div className='lg:w-1/2 flex justify-center
items-start'>
                                    <div className='relative w-full max-w-md
aspect-square bg-black/20 rounded-md shadow-lg overflow-hidden'>
                                          <Image
```

```
                                                src={productUI.Image}
                                                alt={productUI.Name}
                                                fill
                                                style={{ objectFit: 'contain' }}
                                                className='rounded-md'
                                          />
                                    </div>
                              </div>
                        )}

                        {/* Product Information */}
                        <div className='lg:w-1/2'>
                              <ProductInfo product={productUI} />

                              {/* Action Buttons */}
                              <div className='flex flex-wrap gap-4 mt-8'>
                                    <AddtoCart
                                          product={productUI}
                                          className='bg-black/50 hover:bg-
yellow-500'>
                                          Add to Cart
                                    </AddtoCart>
                                    <FavoriteButton product={productUI} />
                                    <SocialButtons product={productUI} />
                              </div>
                        </div>
                  </Section>

                  {/* Related Products */}
                  <RelatedProducts
                        currentProduct={productUI}
                        categoryName={categorySlug}
                        products={relatedProducts}
                  />

                  {/* Comments Section */}

                  <Comments product={productUI} />
            </Main>
      );
}

export default ProductPage;
</file>

<file path="client/src/app/menu/[Products]/components/LikesButton.tsx">
'use client';

import React, { useState, useEffect } from 'react';
import { BiSolidLike } from 'react-icons/bi';
import Button from '@/components/ui/Button';
import { useAuth } from '@/lib/supabase/auth/useAuth';
import { toggleLike } from '@/lib/supabase/likes';
import { createClient } from '@/lib/supabase/client';

interface LikesButtonProps {
      table?: string;
      itemId: string;
      className?: string;
}

const LikesButton: React.FC<LikesButtonProps> = ({ table = 'products', itemId,
className }) => {
      const [likesCount, setLikesCount] = useState(0);
```

```
	const [isLiked, setIsLiked] = useState(false);
	const [loading, setLoading] = useState(true);
	const { user } = useAuth();
	const userId = (user as any)?.id;

	useEffect(() => {
		let mounted = true;
		const load = async () => {
			try {
				const supabase = createClient();
				const { data, error } = await supabase
					.from(table)
					.select('likes')
					.eq('id', itemId)
					.single();

				if (error) {
					console.error('Error fetching likes:', error);
				}

				if (mounted && data) {
					const likesArray = (data as any).likes || [];
					setLikesCount(likesArray.length);
					if (userId) {
						setIsLiked(likesArray.includes(userId));
					}
				}
			} catch (err) {
				console.error(err);
			} finally {
				if (mounted) setLoading(false);
			}
		};
		load();
		return () => { mounted = false; };
	}, [table, itemId, userId]);

	const handleLike = async () => {
		if (!userId) return;

		// Optimistic update
		const newIsLiked = !isLiked;
		const newCount = newIsLiked ? likesCount + 1 : Math.max(0,
likesCount - 1);

		setIsLiked(newIsLiked);
		setLikesCount(newCount);

		try {
			await toggleLike(table, itemId, userId);
		} catch (error) {
			console.error('Error toggling like:', error);
			// Revert
			setIsLiked(!newIsLiked);
			setLikesCount(likesCount);
		}
	};

	return (
		<Button
			disabled={loading || !userId}
			loading={loading}
			variant='icon'
			onClick={handleLike}
```

```
                  className={`${isLiked ? 'bg-amber-500/20 border border-
amber-400/30' : 'bg-white/5 border border-white/10'} ${className || ''}`}
                  aria-label={isLiked ? 'Unlike' : 'Like'}>
                  <BiSolidLike
                        size={20}
                        className={isLiked ? 'text-amber-400' : 'text-white/70'}
                  />
                  <span className='text-xs text-center font-thin truncate font-
sans'>
                        {likesCount} <span className='hidden md:inline-
flex'>Likes</span>
                  </span>
            </Button>
      );
};

export default LikesButton;
</file>

<file path="client/src/app/menu/cart/components/CartFooter.tsx">
'use client';
import Button from '@/components/ui/Button';
import { useCart } from '@/lib/context/CartContext';
import { addOrder } from '@/lib/supabase/orders/orders';
import { useRouter } from 'next/navigation';
import { useState } from 'react';
import { IoAddCircle, IoCheckmarkCircleSharp } from 'react-icons/io5';
import { TbCancel } from 'react-icons/tb';
import { HiShoppingCart } from 'react-icons/hi2';

function CartFooter() {
      const router = useRouter();
      const { cartItems, clearCart, totalPrice, totalQuantity } = useCart();
      const [isPlacingOrder, setIsPlacingOrder] = useState(false);

      const handlePlaceOrder = async () => {
            if (isPlacingOrder) return;
            setIsPlacingOrder(true);
            try {
                  // Get the authenticated user from Supabase
                  const { createClient } = await
import('@/lib/supabase/client');
                  const supabase = createClient();
                  const { data: { user } } = await supabase.auth.getUser();

                  // Only authenticated users can place orders
                  if (!user) {
                        alert('Please log in to place an order.');
                        router.push('/login');
                        return;
                  }

                  await addOrder(user.id, cartItems, totalPrice, totalQuantity);
                  router.push('/orders');
                  clearCart();
            } catch (error) {
                  console.error('Failed to place order: ', error);
                  alert('There was an issue placing your order. Please try
again.');
            } finally {
                  setIsPlacingOrder(false);
            }
      };
```

```
      const handleCancelOrder = () => {
            clearCart();
            router.back();
      };

      return (
            <footer className='fixed bottom-0 left-0 right-0 z-50 backdrop-blur-
xl bg-gradient-to-t from-black/90 via-black/80 to-black/60 border-t border-
white/10 shadow-2xl'>
                  <div className='max-w-7xl mx-auto px-4 sm:px-6 lg:px-8 py-4'>
                        <div className='flex flex-col lg:flex-row items-center
justify-between gap-4'>
                              {/* Order Summary with improved visual hierarchy
*/}
                              <div className='flex items-center gap-6 w-full
lg:w-auto'>
                                    <div className='flex items-center gap-3 bg-
white/5 px-4 py-2.5 rounded-xl border border-white/10'>
                                          <HiShoppingCart className='text-
amber-400 text-2xl' />
                                          <div>
                                                <p className='text-xs text-
yellow-400 uppercase tracking-wider'>Items</p>
                                                <p className='text-xl font-bold
text-white'>{totalQuantity}</p>
                                          </div>
                                    </div>

                                    <div className='flex items-center gap-3 bg-
gradient-to-r from-amber-500/20 to-yellow-500/20 px-6 py-2.5 rounded-xl border
border-amber-400/30'>
                                          <div>
                                                <p className='text-xs text-
amber-300 uppercase tracking-wider'>Total</p>
                                                <p className='text-2xl font-
extrabold text-white tracking-tight'>
                                                      R{totalPrice.toFixed(2)}
                                                </p>
                                          </div>
                                    </div>
                              </div>

                              {/* Action Buttons with improved styling */}
                              <nav className='flex gap-2.5 w-full lg:w-auto'>
                                    {/* Add Items Button */}
                                    <Button
                                          onClick={() => router.back()}
                                          className='flex-1 lg:flex-none bg-
white/5 hover:bg-white/10 border border-white/10 hover:border-white/20
transition-all duration-200'
                                    >
                                          <IoAddCircle size={20}
className='text-amber-400' />
                                          <span className='hidden sm:inline'>Add
Items</span>
                                          <span className='sm:hidden'>Add</span>
                                    </Button>

                                    {/* Place Order Button - only show if cart
has items */}
                                    {cartItems.length > 0 && (
                                          <Button
                                                onClick={handlePlaceOrder}
                                                disabled={isPlacingOrder}
```

```
                                                className='flex-1 lg:flex-none
bg-gradient-to-r from-amber-600 to-yellow-600 hover:from-amber-500 hover:to-
yellow-500 border-0 shadow-lg hover:shadow-amber-500/50 transition-all
duration-200 disabled:opacity-50 disabled:cursor-not-allowed'
                                          >
                                                {isPlacingOrder ? (
                                                      <>
                                                            <div className='w-5
h-5 border-2 border-white/30 border-t-white rounded-full animate-spin'></div>
                                                            <span
className='hidden sm:inline'>Placing Order...</span>
                                                            <span
className='sm:hidden'>Placing...</span>
                                                      </>
                                                ) : (
                                                      <>

      <IoCheckmarkCircleSharp size={20} />
                                                            <span
className='hidden sm:inline'>Place Order</span>
                                                            <span
className='sm:hidden'>Order</span>
                                                      </>
                                                )}
                                          </Button>
                                    )}

                                    {/* Cancel Button */}
                                    <Button
                                          onClick={handleCancelOrder}
                                          className='flex-1 lg:flex-none bg-
red-500/10 hover:bg-red-500/20 border border-red-500/30 hover:border-red-500/50
text-red-300 hover:text-red-200 transition-all duration-200'
                                    >
                                          <TbCancel size={20} />
                                          <span className='hidden
sm:inline'>Cancel</span>
                                    </Button>
                              </nav>
                        </div>
                  </div>
            </footer>
      );
}

export default CartFooter;
</file>

<file path="client/src/app/menu/components/ProductCard.tsx">
import React from 'react';
import Image from 'next/image';
import AppLink from '@/components/ui/Link';
import LikesButton from '@/app/menu/[Products]/components/LikesButton';
import { HiStar } from 'react-icons/hi2';
import AddtoCart from '@/app/menu/[Products]/[product]/components/AddtoCart';

interface ProductCardProps {
      product: ProductType;
      categoryName: string;
}

const ProductCard: React.FC<ProductCardProps> = ({ product, categoryName }) => (
      <article className='m-0 group flex flex-col h-full bg-gradient-to-br from-
yellow-900/80 to-yellow-800/80 backdrop-blur-md border border-white/10 rounded-
```

```
xl p-4 transition-all duration-300 hover:border-amber-400/40 hover:shadow-xl
hover:shadow-amber-500/20'>
            <span className='flex items-center justify-between gap-2 mb-3'>
                  {product.badge && (
                        <span className='px-3 py-1.5 rounded-full text-xs font-
bold z-10 shadow-lg bg-gradient-to-r from-amber-600 to-yellow-600 text-white
uppercase tracking-wider'>
                              {product.badge}
                        </span>
                  )}
                  <LikesButton itemId={product.ProductID} table="products" />
            </span>

            <div className='flex flex-col flex-grow'>
                  <AppLink
                        href={`/menu/${categoryName}/$
{product.Name.toLowerCase().replace(/\s+/g, '-')}`}
                        className='flex flex-col items-center text-center flex-
grow rounded-lg focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-
offset-black/50 focus:ring-amber-500 transition-all duration-200'>
                        {product.Image && (
                              <div className='relative w-full aspect-[4/3] mb-4
bg-black/30 rounded-lg overflow-hidden border border-white/5'>
                                    <Image
                                          src={product.Image}
                                          alt={product.Name}
                                          fill
                                          className='object-contain rounded-lg
transition-transform duration-300 group-hover:scale-110'
                                    />
                              </div>
                        )}
                        <h2 className='text-base font-bold text-white mb-2 line-
clamp-2 flex-grow'>
                              {product.Name}
                        </h2>
                        <p className='text-2xl font-extrabold text-transparent
bg-gradient-to-r from-amber-400 to-yellow-400 bg-clip-text mb-3'>
                              R{product.Price.toFixed(2)}
                        </p>
                        <div className='flex items-center justify-center gap-1
mb-4'>
                              <HiStar className='text-amber-500' size={18} />
                              <HiStar className='text-amber-500' size={18} />
                              <HiStar className='text-amber-500' size={18} />
                              <HiStar className='text-amber-500' size={18} />
                              <HiStar className='text-white/30' size={18} />
                              <span className='text-xs text-yellow-400 font-
semibold ml-2'>(4.5)</span>
                        </div>
                  </AppLink>
            </div>

            <div className='mt-auto'>
                  <AddtoCart product={product}>Add to Cart</AddtoCart>
            </div>
      </article>
);

export default ProductCard;
</file>

<file path="client/src/app/orders/components/OrderItem.tsx">
import React from 'react';
```

```
import { HiClock, HiCurrencyDollar, HiShoppingBag, HiCheckCircle } from 'react-
icons/hi2';

const OrderItem: React.FC<{ order: any }> = ({ order }) => {
      const getStatusColor = (status: string) => {
            switch (status?.toLowerCase()) {
                  case 'completed':
                        return 'bg-green-500/20 text-green-300 border-
green-500/30';
                  case 'processing':
                        return 'bg-amber-500/20 text-amber-300 border-
amber-500/30';
                  case 'cancelled':
                        return 'bg-red-500/20 text-red-300 border-red-500/30';
                  default:
                        return 'bg-blue-500/20 text-blue-300 border-
blue-500/30';
            }
      };

      const getStatusIcon = (status: string) => {
            switch (status?.toLowerCase()) {
                  case 'completed':
                        return <HiCheckCircle className="text-green-400"
size={20} />;
                  case 'processing':
                        return <div className="w-4 h-4 border-2 border-
amber-400/30 border-t-amber-400 rounded-full animate-spin"></div>;
                  default:
                        return <HiClock className="text-blue-400" size={20} />;
            }
      };

      // Parse items from JSON
      const items = Array.isArray(order.items) ? order.items : (typeof
order.items === 'string' ? JSON.parse(order.items) : []);

      return (
            <div className='rounded-xl border border-white/10 bg-gradient-to-br
from-yellow-900/80 to-yellow-800/80 backdrop-blur-sm hover:border-amber-400/30
transition-all duration-300 hover:shadow-lg hover:shadow-amber-500/10 overflow-
hidden'>
                  {/* Order Header */}
                  <div className='bg-gradient-to-r from-amber-500/10 to-
yellow-500/10 border-b border-white/10 px-6 py-4'>
                        <div className='flex flex-wrap items-center justify-
between gap-4'>
                              <div className='flex items-center gap-3'>
                                    <div className='bg-amber-500/20 p-2 rounded-
lg border border-amber-400/30'>
                                          <HiShoppingBag className='text-
amber-400 text-xl' />
                                    </div>
                                    <div>
                                          <p className='text-xs text-yellow-400
uppercase tracking-wider'>Order Date</p>
                                          <p className='text-white font-
semibold'>
                                                {new
Date(order.created_at).toLocaleDateString('en-US', {
                                                      year: 'numeric',
                                                      month: 'short',
                                                      day: 'numeric',
                                                      hour: '2-digit',
```

```
                                                minute: '2-digit'
                                          })}
                                    </p>
                              </div>
                        </div>

                        <div className={`flex items-center gap-2 px-4 py-2
rounded-full border ${getStatusColor(order.status)}`}>
                              {getStatusIcon(order.status)}
                              <span className='font-semibold text-sm
uppercase tracking-wider'>
                                    {order.status || 'pending'}
                              </span>
                        </div>
                  </div>
            </div>

            {/* Order Details */}
            <div className='px-6 py-5 space-y-4'>
                  {/* Order ID */}
                  <div className='bg-white/5 px-4 py-3 rounded-lg border
border-white/10'>
                        <p className='text-xs text-yellow-400 uppercase
tracking-wider mb-1'>Order ID</p>
                        <p className='text-white font-mono text-sm break-
all'>{order.id}</p>
                  </div>

                  {/* Products List */}
                  {items && items.length > 0 && (
                        <div className='bg-white/5 px-4 py-3 rounded-lg
border border-white/10'>
                              <p className='text-xs text-yellow-400
uppercase tracking-wider mb-3'>Items Ordered</p>
                              <div className='space-y-2'>
                                    {items.map((item: any, index: number)
=> (
                                          <div key={index} className='flex
justify-between items-center py-2 border-b border-white/5 last:border-0'>
                                                <div className='flex-1'>
                                                      <p className='text-
white font-medium'>{item.name || item.Name}</p>
                                                      <p className='text-xs
text-yellow-400'>Qty: {item.quantity}</p>
                                                </div>
                                                <p className='text-
amber-400 font-semibold'>
                                                      R{((item.price ||
item.Price) * item.quantity).toFixed(2)}
                                                </p>
                                          </div>
                                    ))}
                              </div>
                        </div>
                  )}

                  {/* Order Summary */}
                  <div className='grid grid-cols-2 gap-3'>
                        <div className='bg-white/5 px-4 py-3 rounded-lg
border border-white/10'>
                              <div className='flex items-center gap-2
mb-1'>
                                    <HiCurrencyDollar className='text-
amber-400' size={16} />
```

```
                                        <p className='text-xs text-yellow-400
uppercase tracking-wider'>Total</p>
                                  </div>
                                  <p className='text-white font-bold text-lg'>
                                        R{order.total_price?.toFixed(2) ||
'0.00'}
                                  </p>
                            </div>

                            <div className='bg-white/5 px-4 py-3 rounded-lg
border border-white/10'>
                                  <div className='flex items-center gap-2
mb-1'>
                                        <HiShoppingBag className='text-
amber-400' size={16} />
                                        <p className='text-xs text-yellow-400
uppercase tracking-wider'>Items</p>
                                  </div>
                                  <p className='text-white font-bold text-lg'>
                                        {order.total_quantity || 0}
                                  </p>
                            </div>
                      </div>
                </div>
           </div>
      );
};

export default OrderItem;
</file>

<file path="client/src/components/ui/Loading.tsx">
import Image from 'next/image';
import '../components.css'; // Optional: for custom styles
import Main from './layout/Main';
import Section from './layout/Section';

const Loading = ({ message = 'Loading...' }: { message?: string }) => (
      <Main className='flex flex-col justify-center items-center min-h-screen
p-4' tittle='PA Luxe Creation' heading='Experience the unique convenience of
delicious food and a state-of-the-art Photo boot, all in
                        one place in Lulekani.'>
           <Section className='flex flex-col items-center text-center max-w-
md'>
                <div className='mt-8 animate-pulse'>
                      <Image
                            src='/PA_Logo.png'
                            alt='PA Luxe Creation Logo'
                            width={220}
                            height={220}
                            className='w-48 h-48 mb-4'
                            priority
                      />
                </div>
                <div className='spinner' />
                <p className='text-white font-semibold'>{message}</p>
           </Section>
      </Main>
);

export default Loading;
</file>

<file path="client/src/lib/forms/EditProfileForm.tsx">
```

```
'use client';

import React, { useState, useEffect } from 'react';
import { useRouter } from 'next/navigation';
import { useActionState } from 'react';
import { FaSave, FaTimes, FaUserCircle } from 'react-icons/fa';
import TextInput from '@/components/ui/TextInput';
import Button from '@/components/ui/Button';
import Avatar from '@/components/ui/Avatar';
import Loading from '@/components/ui/Loading';
import { ImageUpload } from '@/components/ui/ImageUpload';
import { useAuth } from '@/lib/supabase/auth/useAuth';
import { updateProfileAction } from '@/lib/actions/profile';

import Section from '@/components/ui/layout/Section';

const EditProfileForm: React.FC = () => {
    const { user, loading } = useAuth();
    const router = useRouter();
    const [state, formAction, isPending] = useActionState(updateProfileAction,
null);

    // State for form fields
    const [displayName, setDisplayName] = useState('');
    const [phoneNumber, setPhoneNumber] = useState('');
    const [photoURL, setPhotoURL] = useState('');

    // Validation states
    const [displayNameError, setDisplayNameError] = useState(false);
    const [phoneNumberError, setPhoneNumberError] = useState(false);

    // Effect to initialize form fields when user data loads or changes
    useEffect(() => {
        if (user) {
            setDisplayName(user.user_metadata?.displayName || '');
            setPhoneNumber(user.user_metadata?.phoneNumber || '');
            setPhotoURL(user.user_metadata?.photoURL || '');
        }
    }, [user]);

    // Handle redirection if user is not logged in after loading
    useEffect(() => {
        if (!loading && !user) {
            router.replace('/login');
        }
    }, [user, loading, router]);

    // Redirect on success
    useEffect(() => {
        if (state?.success) {
            router.push('/profile');
        }
    }, [state?.success, router]);

    // Show loading state while user data is being fetched by context
    if (loading) {
        return <Loading message='Loading profile data...' />;
    }

    // If user is null after loading, it means redirection is happening
    if (!user) {
        return null;
    }
```

```
    const validateForm = () => {
        let isValid = true;

        if (!displayName.trim()) {
            setDisplayNameError(true);
            isValid = false;
        } else {
            setDisplayNameError(false);
        }

        // Basic phone number validation
        if (phoneNumber.trim() && !/^\+?[0-9\s-()]
{7,20}$/.test(phoneNumber.trim())) {
            setPhoneNumberError(true);
            isValid = false;
        } else {
            setPhoneNumberError(false);
        }

        return isValid;
    };

    const handleSubmit = (e: React.FormEvent<HTMLFormElement>) => {
        if (!validateForm()) {
            e.preventDefault();
            return;
        }
    };

    const handleCancel = () => {
        router.back();
    };

    return (
        <Section tittle="Edit Profile" Icon={FaUserCircle} heading="Update your
profile details below." >
            <div className='flex flex-col items-center mb-8'>
                <Avatar
                    src={photoURL || user.user_metadata?.photoURL || ''}
                    alt={displayName || user.user_metadata?.displayName ||
user.email || 'User'}
                    size='large'
                    className='mb-4 shadow-md'
                />
                <h2 className='text-yellow-500 text-3xl font-bold mb-2'>Edit
Personal Information</h2>
                <p className='text-yellow-500 text-lg'>Update your profile
details below.</p>
            </div>

            {state?.error && (
                <div className='text-yellow-500 mb-4 bg-yellow-900/20 p-3
rounded-md'>{state.error}</div>
            )}
            {state?.success && (
                <div className='text-white mb-4 bg-green-900/20 p-3 rounded-
md'>Profile updated successfully!</div>
            )}

            <form
                action={formAction}
                onSubmit={handleSubmit}
                className='space-y-6'>
                {/* Display Name */}
```

```
                <div>
                    <TextInput
                        type='text'
                        label='Display Name'
                        name='displayName'
                        placeholder='Your Name'
                        value={displayName}
                        onChange={(e) => {
                            setDisplayName(e.target.value);
                            setDisplayNameError(false);
                        }}
                        variant={displayNameError ? 'error' : 'default'}
                        required
                    />
                    {displayNameError && (
                        <p className='text-yellow-500 text-sm mt-1'>Display Name
is required.</p>
                    )}
                </div>

                {/* Email (Read-only) */}
                <div>
                    <TextInput
                        type='email'
                        label='Email (Cannot be changed here)'
                        value={user.email || ''}
                        disabled
                        className='cursor-not-allowed bg-black/50'
                    />
                    <p className='text-sm mt-1'>
                        To change your email, please visit your account settings
or contact support.
                    </p>
                </div>

                {/* Phone Number */}
                <div>
                    <TextInput
                        type='tel'
                        label='Phone Number'
                        name='phoneNumber'
                        placeholder='e.g., +27 12 345 6789'
                        value={phoneNumber}
                        onChange={(e) => {
                            setPhoneNumber(e.target.value);
                            setPhoneNumberError(false);
                        }}
                        variant={phoneNumberError ? 'error' : 'default'}
                    />
                    {phoneNumberError && (
                        <p className='text-yellow-500 text-sm mt-1'>
                            Please enter a valid phone number (optional).
                        </p>
                    )}
                </div>

                {/* Profile Photo */}
                <div>
                    <label className='block text-sm font-medium mb-2 text-
yellow-500'>Profile Photo</label>
                    <ImageUpload
                        defaultValue={photoURL}
                        onChange={(url) => setPhotoURL(url)}
                        folder='profiles'
```

```
                        label='Upload Profile Photo'
                    />
                    <p className='text-sm mt-2 text-white/70'>Upload a new
profile picture</p>
                </div>

                {/* Action Buttons */}
                <div className='flex flex-col sm:flex-row gap-4 justify-end
pt-4'>
                    <Button
                        type='button'
                        onClick={handleCancel}
                        variant='secondary'
                        size='md'
                        className='w-full sm:w-auto'>
                        <FaTimes className='mr-2' /> Cancel
                    </Button>
                    <Button
                        type='submit'
                        variant='primary'
                        size='md'
                        loading={isPending}
                        disabled={isPending}
                        className='w-full sm:w-auto'>
                        <FaSave className='mr-2' /> {isPending ? 'Saving...' :
'Save Changes'}
                    </Button>
                </div>
            </form>
        </Section>
    );
};

export default EditProfileForm;
</file>

<file path="client/src/lib/forms/LoginForm.tsx">
'use client';

import { useActionState } from 'react';
import { signInAction } from '@/lib/actions/auth';
import Loading from '@/components/ui/Loading';

import Button from '@/components/ui/Button';
import TextInput from '@/components/ui/TextInput';
import AppLink from '@/components/ui/Link';
import Section from '@/components/ui/layout/Section';

export default function LoginForm() {
    const [state, formAction, isPending] = useActionState(signInAction, null);

    if (isPending) {
        return <Loading message="Signing in..." />;
    }

    return (
        <Section >
            {state?.error && (
                <div className='text-red-300 mb-4 mt-4 bg-red-500/10 border
border-red-500/30 p-3 rounded-lg'>{state.error}</div>
            )}
            <form
                action={formAction}
                className='space-y-6 mt-6'>
```

```
                <TextInput
                    label='Email:'
                    type='email'
                    id='email'
                    name='email'
                    required
                />
                <TextInput
                    label='Password:'
                    type='password'
                    id='password'
                    name='password'
                    required
                />
                <Button
                    type='submit'
                    variant='primary'
                    className='w-full'
                    size='lg'
                    loading={isPending}>
                    Sign In
                </Button>
            </form>
            <p className='mt-6 text-white/70'>
                Don&apos;t have an account? <AppLink href='/register'>Sign
Up</AppLink>
            </p>
        </Section>
    );
}
</file>

<file path="client/src/lib/forms/RegisterForm.tsx">
'use client';

import { useActionState } from 'react';
import { signUpAction } from '@/lib/actions/auth';
import Loading from '@/components/ui/Loading';
import Button from '@/components/ui/Button';
import TextInput from '@/components/ui/TextInput';
import AppLink from '@/components/ui/Link';
import Section from '@/components/ui/layout/Section';

const RegisterForm = () => {
    const [state, formAction, isPending] = useActionState(signUpAction, null);

    if (isPending) {
        return <Loading message="Registering..." />;
    }

    return (
        <Section >
            {state?.error && (
                <div className='text-red-300 mb-4 mt-4 bg-red-500/10 border
border-red-500/30 p-3 rounded-lg'>{state.error}</div>
            )}
            <form
                action={formAction}
                className='space-y-6 mt-6'>
                <TextInput
                    label='Email'
                    id='email'
                    name='email'
                    type='email'
```

```
                    placeholder='Email'
                    required
                />
                <TextInput
                    label='Display Name'
                    id='displayName'
                    name='displayName'
                    type='text'
                    placeholder='Display Name'
                    required
                />
                <TextInput
                    label='Phone Number'
                    id='phoneNumber'
                    name='phoneNumber'
                    type='tel'
                    placeholder='Phone Number'
                    required
                />
                <TextInput
                    label='Password'
                    id='password'
                    name='password'
                    type='password'
                    placeholder='******************'
                    required
                />
                <Button
                    type='submit'
                    variant='primary'
                    className='w-full'
                    size='lg'
                    loading={isPending}>
                    Register
                </Button>
            </form>
            <p className='text-center text-sm mt-6 text-white/70'>
                Already have an account? <AppLink href='/login'>Login</AppLink>
            </p>
        </Section>
    );
};

export default RegisterForm;
</file>

<file path="client/src/app/contact/page.tsx">
import ContactForm from '@/lib/forms/ContactForm';
import ContactInfo from './components/ContactInfo';
import MapEmbed from './components/MapEmbed';


export default function ContactPage() {
  return (
    <div className="py-12 px-4 sm:px-6 lg:px-8 max-w-7xl mx-auto">
      <div className="text-center mb-12">
        <h1 className="text-4xl font-bold text-white mb-4">Get In Touch</h1>
        <div className="w-24 h-1 bg-yellow-500/95 mx-auto"></div>
        <p className="mt-6 text-lg text-gray-300 max-w-2xl mx-auto">
          Have questions or feedback? We'd love to hear from you. Fill out the
form below or reach out to us directly.
        </p>
      </div>
```

```
      <ContactInfo />

      <div className="mt-16">
        <h2 className="text-3xl font-bold text-white mb-8 text-center">Send Us a
Message</h2>
        <ContactForm />
      </div>

      <div className="mt-16">
        <h2 className="text-3xl font-bold text-white mb-8 text-center">Find Us
on the Map</h2>
        <MapEmbed />
      </div>
    </div>
  );
}
</file>

<file
path="client/src/app/menu/[Products]/[product]/components/FavoriteButton.tsx">
'use client';

import React, { useState, useEffect } from 'react';
import { FaHeart, FaRegHeart } from 'react-icons/fa';
import { useAuth } from '@/lib/supabase/auth/useAuth';
import { getUserFavorites, toggleFavorite } from '@/lib/supabase/favorites';

import Button from '@/components/ui/Button';

interface FavoriteButtonProps {
      product: ProductType;
}

const FavoriteButton: React.FC<FavoriteButtonProps> = ({ product }) => {
      const [isFavorite, setIsFavorite] = useState(false);
      const [isLoading, setIsLoading] = useState(false);
      const { user } = useAuth();
      const userId = (user as any)?.id || (user as any)?.uid || null;

      // Determine product id/key used in this app (fallbacks for different
shapes)
      const productKey = (product as any).ProductID || (product as any).id ||
(product as any).Name || (product as any).slug || (product as any).Slug;

      useEffect(() => {
            if (!userId || !productKey) return;

            const checkFavoriteStatus = async () => {
                  try {
                        const favorites = await getUserFavorites(userId);
                        setIsFavorite(favorites.includes(productKey));
                  } catch (error) {
                        console.error('Error checking favorite status:', error);
                  }
            };

            checkFavoriteStatus();
      }, [userId, productKey]);

      const handleToggleFavorite = async () => {
            if (!userId) {
                  alert('Please log in to save favorites');
                  return;
            }
```

```
            setIsLoading(true);
            try {
                  const result = await toggleFavorite(userId, productKey);
                  setIsFavorite(result.added);
            } catch (error) {
                  console.error('Error toggling favorite:', error);
                  alert('Failed to update favorites');
            } finally {
                  setIsLoading(false);
            }
      };

      return (
            <Button
                  onClick={handleToggleFavorite}
                  disabled={isLoading || !userId}
                  className={`flex items-center justify-center gap-2 px-4 py-2.5
rounded-xl border font-semibold text-white transition-all duration-200
focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-offset-black/50
focus:ring-amber-500 shadow-md ${isLoading || !userId
                        ? 'bg-yellow-700/50 border-white/10 cursor-not-allowed
opacity-50'
                        : 'bg-gradient-to-r from-amber-600 to-yellow-600
hover:from-amber-500 hover:to-yellow-500 border-0 hover:shadow-amber-500/50'
                        }`}
                  aria-live='polite'>
                  {isLoading ? (
                        <>
                              <div className='w-4 h-4 border-2 border-white/50
border-t-white rounded-full animate-spin'></div>
                              <span>Updating...</span>
                        </>
                  ) : (
                        <>
                              {isFavorite ? <FaHeart /> : <FaRegHeart />}
                              <span>{isFavorite ? 'Saved' : 'Save to Favorites'}
</span>
                        </>
                  )}
            </Button>
      );
};

export default FavoriteButton;
</file>

<file path="client/src/app/menu/[Products]/components/FilterSortBar.tsx">
'use client';

import React, { useState, useEffect } from 'react';
import { FaSort, FaFilter } from 'react-icons/fa';
import Button from '@/components/ui/Button';
import { useRouter, useSearchParams, usePathname } from 'next/navigation';

const FilterSortBar: React.FC = () => {
      const router = useRouter();
      const searchParams = useSearchParams();
      const pathname = usePathname();

      const currentSort = searchParams.get('sort') || 'default';
      const currentMinPrice = Number(searchParams.get('minPrice')) || 0;
      const currentMaxPrice = Number(searchParams.get('maxPrice')) || 1000;
```

```
      const [showFilters, setShowFilters] = useState(false);
      const [localMinPrice, setLocalMinPrice] = useState(currentMinPrice);
      const [localMaxPrice, setLocalMaxPrice] = useState(currentMaxPrice);

      useEffect(() => {
            setLocalMinPrice(currentMinPrice);
            setLocalMaxPrice(currentMaxPrice);
      }, [currentMinPrice, currentMaxPrice]);

      const updateParams = (updates: Record<string, string | number | null>) =>
{
            const params = new URLSearchParams(searchParams.toString());
            Object.entries(updates).forEach(([key, value]) => {
                  if (value === null) {
                        params.delete(key);
                  } else {
                        params.set(key, String(value));
                  }
            });
            router.replace(`${pathname}?${params.toString()}`, { scroll:
false });
      };

      const handleSortChange = (sortType: string) => {
            updateParams({ sort: sortType });
      };

      const handleApplyFilters = () => {
            updateParams({
                  minPrice: localMinPrice,
                  maxPrice: localMaxPrice,
            });
            setShowFilters(false);
      };

      const handleResetFilters = () => {
            setLocalMinPrice(0);
            setLocalMaxPrice(1000);
            updateParams({
                  minPrice: null,
                  maxPrice: null,
                  sort: null,
            });
            setShowFilters(false);
      };

      const inputStyles =
            'w-24 p-2 border border-amber-400/30 rounded-md bg-yellow-900/50
text-amber-100 placeholder-amber-400/50 focus:outline-none focus:ring-2
focus:ring-amber-400/50 focus:border-amber-400 transition-all duration-200';

      const buttonStyles = 'bg-gradient-to-r from-amber-600/20 to-yellow-600/20
border border-amber-400/30 text-amber-400 hover:bg-amber-600/30 hover:text-
amber-300 transition-all duration-200';

      return (
            <div className='bg-gradient-to-br from-yellow-900/20 to-amber-900/20
backdrop-blur-md p-3 rounded-b-xl border border-amber-400/10 sticky top-20 z-20
col-span-full mt-0 shadow-xl shadow-black/20'>
                  <div className='flex flex-col md:flex-row gap-4 items-center
justify-between'>
                        {/* Sort Options */}
                        <div className='flex items-center gap-3'>
                              <div className='p-2 rounded-lg bg-amber-400/10
```

```
text-amber-400'>
                                    <FaSort size={16} />
                              </div>
                              <label
                                    htmlFor='sort-select'
                                    className='font-semibold text-amber-100'>
                                    Sort by:
                              </label>
                              <select
                                    id='sort-select'
                                    value={currentSort}
                                    onChange={(e) =>
handleSortChange(e.target.value)}
                                    className='bg-yellow-900/50 text-amber-100
border border-amber-400/30 rounded-lg px-4 py-2 focus:outline-none focus:ring-2
focus:ring-amber-400/50 cursor-pointer hover:border-amber-400/50 transition-all
duration-200'>
                                    <option value='default'>Default</option>
                                    <option value='price-low'>Price: Low to
High</option>
                                    <option value='price-high'>Price: High to
Low</option>
                                    <option value='popularity'>Most
Popular</option>
                                    <option value='name'>Name: A-Z</option>
                              </select>
                        </div>

                        {/* Filter Toggle */}
                        <button
                              onClick={() => setShowFilters(!showFilters)}
                              className={`flex items-center gap-2 px-4 py-2
rounded-lg font-medium transition-all duration-200 ${showFilters ? 'bg-amber-400
text-yellow-900' : buttonStyles}`}>
                              <FaFilter />
                              <span>{showFilters ? 'Hide' : 'Show'}
Filters</span>
                        </button>
                  </div>

                  {/* Filter Panel */}
                  {showFilters && (
                        <article className='mt-4 p-4 rounded-lg bg-yellow-900/50
border border-amber-400/10 space-y-4 animate-in fade-in slide-in-from-top-2
duration-200'>
                              <h3 className='text-lg font-semibold mb-2 text-
amber-400'>Price Range</h3>
                              <div className='flex flex-col md:flex-row flex-
wrap gap-4 items-center'>
                                    <div className='flex items-center gap-2'>
                                          <label
                                                htmlFor='min-price'
                                                className='text-sm text-
amber-200'>
                                                Min:
                                          </label>
                                          <input
                                                id='min-price'
                                                type='number'
                                                value={localMinPrice}
                                                onChange={(e) =>
setLocalMinPrice(Number(e.target.value))}
                                                className={inputStyles}
                                                placeholder='0'
```

```
                                    />
                              </div>
                              <div className='flex items-center gap-2'>
                                    <label
                                          htmlFor='max-price'
                                          className='text-sm text-
amber-200'>
                                          Max:
                                    </label>
                                    <input
                                          id='max-price'
                                          type='number'
                                          value={localMaxPrice}
                                          onChange={(e) =>
setLocalMaxPrice(Number(e.target.value))}
                                          className={inputStyles}
                                          placeholder='1000'
                                    />
                              </div>
                              <div className='flex gap-2 ml-auto'>
                                    <button
                                          onClick={handleResetFilters}
                                          className='px-4 py-2 rounded-lg
text-sm font-medium text-yellow-400 hover:text-white hover:bg-white/5
transition-colors'>
                                          Reset
                                    </button>
                                    <button
                                          onClick={handleApplyFilters}
                                          className='px-6 py-2 rounded-lg
text-sm font-bold bg-amber-400 text-yellow-900 hover:bg-amber-300 shadow-lg
shadow-amber-400/20 transition-all duration-200'>
                                          Apply Filters
                                    </button>
                              </div>
                        </div>
                  </article>
            )}
      </div>
  );
};

export default FilterSortBar;
</file>

<file path="client/src/app/orders/layout.tsx">
'use client';

import { CartProvider } from '@/lib/context/CartContext';
import React, { ReactNode } from 'react';
import Link from 'next/link';
import { HiShoppingBag, HiClock, HiCog, HiCheckCircle, HiArrowLeft, HiXCircle }
from 'react-icons/hi2';
import { usePathname } from 'next/navigation';

interface LayoutProps {
      children: ReactNode;
}

const Layout: React.FC<LayoutProps> = ({ children }) => {
      const pathname = usePathname();

      const navItems = [
            { href: '/orders/pending', label: 'Pending', icon: HiClock },
```

```
            { href: '/orders/processing', label: 'Processing', icon: HiCog },
            { href: '/orders/completed', label: 'Completed', icon: HiCheckCircle
},
            { href: '/orders/cancelled', label: 'Cancelled', icon: HiXCircle },
      ];

      return (
            <CartProvider>
                  {/* Enhanced Navigation */}
                  <nav className='sticky top-[73px] bg-gradient-to-r from-
yellow-900/98 to-yellow-800/98 backdrop-blur-xl border-b border-amber-400/10
shadow-lg z-40 px-4 py-3'>
                        <div className='max-w-7xl mx-auto flex items-center
gap-4'>
                              {/* Back to Menu Button */}
                              <Link
                                    href='/menu'
                                    className='flex items-center gap-2 px-4 py-2
rounded-lg text-yellow-300 hover:text-white hover:bg-white/5 border border-
transparent hover:border-white/10 transition-all duration-200 font-medium'>
                                    <HiArrowLeft className='text-lg' />
                                    <span className='hidden
sm:inline'>Menu</span>
                              </Link>

                              {/* Divider */}
                              <div className='h-8 w-px bg-white/10'></div>

                              {/* Status Filter Tabs */}
                              <div className='flex gap-2 flex-1 overflow-x-
auto'>
                                    {navItems.map(({ href, label, icon: Icon })
=> {
                                          const isActive = pathname === href;
                                          return (
                                                <Link
                                                      key={href}
                                                      href={href}
                                                      className={`flex items-
center gap-2 px-4 py-2 rounded-lg font-medium transition-all duration-200
whitespace-nowrap ${isActive
                                                            ? 'bg-gradient-to-r
from-amber-600/20 to-yellow-600/20 border border-amber-400/30 text-amber-400
shadow-md shadow-amber-500/10'
                                                            : 'text-yellow-300
hover:text-white hover:bg-white/5 border border-transparent
hover:border-white/10'
                                                      }`}>
                                                      <Icon className={`text-lg $
{isActive ? 'text-amber-400' : ''}`} />
                                                      <span className='text-sm
sm:text-base'>{label}</span>
                                                </Link>
                                          );
                                    })}
                              </div>

                              {/* Orders Icon (decorative) */}
                              <div className='hidden md:flex items-center gap-2
text-yellow-400'>
                                    <HiShoppingBag className='text-xl text-
amber-400' />
                                    <span className='text-sm'>Orders</span>
                              </div>
```

```
                        </div>
                  </nav>

                  {children}
            </CartProvider>
      );
};

export default Layout;
</file>

<file path="client/src/app/profile/edit/page.tsx">
import Main from "@/components/ui/layout/Main";
import EditProfileForm from "@/lib/forms/EditProfileForm";


const EditProfilePage: React.FC = () => {

      return (
            <Main tittle="Edit Profile">
                  <EditProfileForm />
            </Main>
      );
};

export default EditProfilePage;
</file>

<file path="client/src/components/Navbar/AuthButton.tsx">
'use client';
import AppLink from '@/components/ui/Link';
import Button from '@/components/ui/Button';
import { HiLogin, HiUserCircle } from 'react-icons/hi';
import { User } from '@supabase/supabase-js';

const AuthButton: React.FC<{ setMenubar: (path: 'mobile' | 'profile') => void;
user: User | null }> = ({
      setMenubar,
      user,
}) => {
      if (user)
            return (
                  <Button
                        variant='icon'
                        onClick={() => setMenubar('profile')}
                        className='bg-white/5 hover:bg-amber-500/20 border
border-white/10 hover:border-amber-400/30 transition-all duration-200 p-2
rounded-lg'
                        aria-label='Open profile menu'>
                        <HiUserCircle size={28} className='text-amber-400' />
                  </Button>
            );
      return (
            <AppLink
                  href='/login'
                  className='flex items-center gap-2 px-4 py-2 bg-gradient-to-r
from-amber-600 to-yellow-600 hover:from-amber-500 hover:to-yellow-500 rounded-lg
font-semibold text-white shadow-lg hover:shadow-amber-500/50 transition-all
duration-200 border-0'>
                  <HiLogin size={20} />
                  <span className='hidden sm:inline'>Log in</span>
            </AppLink>
      );
};
```

```
export default AuthButton;
</file>

<file path="admin/src/lib/data/products-actions.ts">
"use server";

import { revalidatePath } from "next/cache";
import { createSupabaseServerClient } from "@/lib/supabase/server";
import type { Database } from "@/lib/database.types";

export type ProductActionState = {
  error?: string;
  success?: string;
};

const sanitize = (value: FormDataEntryValue | null) =>
  String(value ?? "").trim();

export const createProductAction = async (
  _prev: ProductActionState,
  formData: FormData,
): Promise<ProductActionState> => {
  console.log("createProductAction: Received formData");
  const name = sanitize(formData.get("name"));
  const category = sanitize(formData.get("category"));
  const description = sanitize(formData.get("description"));
  const badge = sanitize(formData.get("badge")) || null;
  const imageUrl = sanitize(formData.get("image_url")) || null;
  console.log("createProductAction: Image URL:", imageUrl);
  const price = Number(formData.get("price") ?? 0);
  const stock = Number(formData.get("stock") ?? 0);

  if (!name || !category) {
    return { error: "Name and category are required." };
  }

  if (Number.isNaN(price) || price <= 0) {
    return { error: "Price must be greater than zero." };
  }

  const supabase = await createSupabaseServerClient();
  const productData: Database["public"]["Tables"]["products"]["Insert"] = {
    name,
    category_name: category,
    description: description || null,
    badge,
    image_url: imageUrl,
    price,
    stock: Number.isNaN(stock) ? 0 : stock,
  };
  console.log("createProductAction: Inserting data:", productData);
  const { error } = await supabase.from("products").insert(productData as
never);

  if (error) {
    console.error("Failed to create product", error);
    return { error: error.message };
  }

  revalidatePath("/menu");
  revalidatePath("/products");
  return { success: `${name} added to menu.` };
};
```

```
export const updateProductAction = async (
  _prev: ProductActionState,
  formData: FormData,
): Promise<ProductActionState> => {
  console.log("updateProductAction: Received formData");
  const id = sanitize(formData.get("id"));
  const name = sanitize(formData.get("name")) || undefined;
  const category = sanitize(formData.get("category")) || undefined;
  const priceValue = formData.get("price");
  const stockValue = formData.get("stock");
  const badge = sanitize(formData.get("badge")) || null;
  const description = sanitize(formData.get("description")) || null;
  const imageUrl = sanitize(formData.get("image_url")) || null;
  console.log("updateProductAction: Image URL:", imageUrl);
  const isHidden = formData.get("is_hidden") === "on";

  if (!id) {
    return { error: "Missing product identifier." };
  }

  const price = priceValue !== null ? Number(priceValue) : undefined;
  const stock = stockValue !== null ? Number(stockValue) : undefined;

  const payload: Record<string, unknown> = {
    badge,
    description,
    image_url: imageUrl,
    is_hidden: isHidden,
  };

  if (name) {
    payload.name = name;
  }

  if (category) {
    payload.category_name = category;
  }

  if (price !== undefined && !Number.isNaN(price)) {
    payload.price = price;
  }

  if (stock !== undefined && !Number.isNaN(stock)) {
    payload.stock = stock;
  }

  const supabase = await createSupabaseServerClient();
  const updateData: Database["public"]["Tables"]["products"]["Update"] = {
    ...payload,
  };
  const { error } = await supabase
    .from("products")
    .update(updateData as never)
    .eq("id", id);

  if (error) {
    console.error("Failed to update product", error);
    return { error: error.message };
  }

  revalidatePath("/menu");
  revalidatePath("/products");
  return { success: "Product updated." };
```

```
};

export const deleteProductAction = async (
  productId: string,
): Promise<ProductActionState> => {
  if (!productId) {
    return { error: "Product ID is required." };
  }

  const supabase = await createSupabaseServerClient();
  const { error } = await supabase.from("products").delete().eq("id",
productId);

  if (error) {
    console.error("Failed to delete product", error);
    return { error: error.message };
  }

  revalidatePath("/menu");
  revalidatePath("/products");
  return { success: "Product deleted successfully." };
};
</file>

<file path="client/src/app/home-components/Testimonials.tsx">
'use client';
import React, { useState, useEffect } from 'react';
import { BiSolidComment, BiSolidShare } from 'react-icons/bi';
import { FaQuoteLeft } from 'react-icons/fa';
import Button from '@/components/ui/Button';
import Section from '@/components/ui/layout/Section';
import { useAuth } from '@/lib/supabase/auth/useAuth';
import { useRouter } from 'next/navigation';
import LikesButton from '@/app/menu/[Products]/components/LikesButton';
import { createClient } from '@/lib/supabase/client';

interface Testimonial {
      id: string;
      text: string;
      author: string;
      rating: number;
      likes: string[] | null;
      comments: any[] | null;
}

const Testimonials: React.FC = () => {
      const { user } = useAuth();
      const router = useRouter();
      const [testimonials, setTestimonials] = useState<Testimonial[]>([]);

      useEffect(() => {
            const fetchTestimonials = async () => {
                  const supabase = createClient();
                  const { data, error } = await supabase
                        .from('testimonials')
                        .select('*')
                        .order('created_at', { ascending: false });

                  if (error) {
                        console.error('Error fetching testimonials:', error);
                  } else {
                        setTestimonials(data || []);
                  }
            };
```

```
            fetchTestimonials();
      }, []);

      const handleComment = (testimonialId: string) => {
            if (!user) {
                  router.push('/login');
                  return;
            }
            alert('Comment functionality - would open a modal or navigate to
comments');
      };

      const handleShare = (testimonialId: string) => {
            if (navigator.share) {
                  navigator.share({
                        title: 'PA Luxe Creation Testimonial',
                        text: 'Check out this customer review!',
                        url: window.location.href,
                  });
            } else {
                  alert('Share functionality - URL copied to clipboard');
            }
      };

      return (
            <Section
                  Icon={FaQuoteLeft}
                  tittle='What Our Customers Say'>
                  <div className='grid grid-cols-1 md:grid-cols-2 gap-2'>
                        {testimonials.map((testimonial) => (
                              <article
                                    key={testimonial.id}
                                    className=' rounded-xl border border-
yellow-500/30 bg-gradient-to-br from-yellow-900/20 to-amber-900/20 backdrop-
blur-md hover:border-yellow-400/50 transition-all duration-300 shadow-lg p-6
flex flex-col'>
                                    <div className='flex-grow'>
                                          <FaQuoteLeft className='text-2xl text-
amber-400/50 mb-4' />
                                          <p className='italic text-yellow-300
mb-4'>{testimonial.text}</p>
                                    </div>
                                    <div className='mt-auto'>
                                          <div className='flex items-center
mb-4'>
                                                <span className='flex-shrink-0
w-10 h-10 rounded-full bg-gradient-to-br from-amber-500 to-yellow-500 flex
items-center justify-center text-white font-bold text-lg shadow-md'>

      {testimonial.author.charAt(0).toUpperCase()}
                                                </span>
                                                <p className='font-semibold text-
white ml-3'>{testimonial.author}</p>
                                          </div>
                                          <div className='flex justify-around
items-center border-t border-white/20 pt-4 bg-white/5 -mx-6 px-6 -mb-6 pb-6
rounded-b-xl'>
                                                <div className="flex flex-col
items-center gap-1">
                                                      <LikesButton
itemId={testimonial.id} table="testimonials" />
                                                </div>
                                                <Button
                                                      variant='icon'
```

```
                                                onClick={() =>
handleComment(testimonial.id)}
                                                className='flex flex-col
items-center gap-1'
                                                aria-label={`Comment on
testimonial by ${testimonial.author}`}>
                                                <BiSolidComment
                                                      size={20}
                                                      className='text-white
hover:text-yellow-500 transition-colors'
                                                />
                                                <span className='text-xs
text-yellow-400'>{(testimonial.comments as any[])?.length || 0}</span>
                                          </Button>
                                          <Button
                                                variant='icon'
                                                onClick={() =>
handleShare(testimonial.id)}
                                                className='flex flex-col
items-center gap-1'
                                                aria-label={`Share
testimonial by ${testimonial.author}`}>
                                                <BiSolidShare
                                                      size={20}
                                                      className='text-white
hover:text-yellow-500 transition-colors'
                                                />
                                                <span className='text-xs
text-yellow-400'>Share</span>
                                          </Button>
                                    </div>
                              </div>
                        </article>
                  ))}
            </div>
      </Section>
  );
};

export default Testimonials;
</file>

<file path="client/src/app/menu/[Products]/[product]/comments/page.tsx">
import { createClient } from "@/lib/supabase/server";
import CommentsForm from "@/lib/forms/CommentsForm";
import { redirect } from "next/navigation";
interface CommentsProps {
      params: Promise<{ product: string }>;
}
const Comments:React.FC<CommentsProps> = async ({ params }) => {
      const supabase = await createClient();
      const { data: { user }, error } = await supabase.auth.getUser();
  if (error || !user?.id) {
      return redirect('/login');
  }
  const ProductName = (await params).product;
  return (
      <CommentsForm ProductName={ProductName}  />
  );
};

export default Comments;
</file>
```

<file path="client/src/app/menu/[Products]/page.tsx">
```
import { Suspense } from 'react';
import Loading from '@/components/ui/Loading';
import FilterSortBar from '@/app/menu/[Products]/components/FilterSortBar';
import Main from '@/components/ui/layout/Main';
import ProductCard from '../components/ProductCard';
import { createClient } from '@/lib/supabase/server';

export default async function Page({
      params,
      searchParams,
}: {
      params: Promise<{ Products: string }>;
      searchParams: Promise<{ sort?: string; minPrice?: string; maxPrice?:
string }>;
}) {
      const { Products: categorySlug } = await params;
      const { sort = 'default', minPrice = '0', maxPrice = '1000' } = await
searchParams;
      const supabase = await createClient();

      // Reconstruct category name from slug (e.g., "kota-meals" -> "kota
meals")
      // This is a simple heuristic. For robust matching, we might need a slug
column in DB.
      const categoryName = categorySlug.replace(/-/g, ' ');

      // Fetch products matching the category (case-insensitive)
      const { data: productsData, error } = await supabase
            .from('products')
            .select('*')
            .ilike('category_name', categoryName);

      if (error) {
            console.error('Error fetching products:', error);
      }

      type ProductRow = {
            id: string;
            name: string;
            description: string | null;
            price: number | null;
            image_url: string | null;
            badge: string | null;
            likes: number | null;
      };

      let products: ProductRow[] = (productsData as ProductRow[]) || [];

      // Map to UI ProductType
      let uiProducts: ProductType[] = products.map((p) => ({
            ProductID: p.id,
            Name: p.name,
            Description: [p.description || ''],
            Price: p.price || 0,
            Image: p.image_url || '/Menus/placeholder.png',
            badge: p.badge || undefined,
            rating: p.likes ? 5 : undefined,
      }));

      // Filter by price
      const min = Number(minPrice);
      const max = Number(maxPrice);
      uiProducts = uiProducts.filter((p) => p.Price >= min && p.Price <= max);
```

```
	// Sort products
	switch (sort) {
		case 'price-low':
			uiProducts.sort((a, b) => a.Price - b.Price);
			break;
		case 'price-high':
			uiProducts.sort((a, b) => b.Price - a.Price);
			break;
		case 'popularity':
			// Ideally fetch likes count from DB and sort there, or sort here if we have the data
			// For now, we can't easily sort by popularity without likes data joined.
			// Assuming 'likes' column in products table is the count.
			// If we mapped it to rating, we can use that, or fetch likes separately.
			// Let's assume p.likes is the count.
			uiProducts.sort((a, b) => {
				const likesA = products.find((p: ProductRow) => p.id === a.ProductID)?.likes || 0;
				const likesB = products.find((p: ProductRow) => p.id === b.ProductID)?.likes || 0;
				return likesB - likesA;
			});
			break;
		case 'name':
			uiProducts.sort((a, b) => a.Name.localeCompare(b.Name));
			break;
		default:
			break;
	}

	return (
		<>
			<Suspense fallback={<div className="h-12" />}>
				<FilterSortBar />
			</Suspense>
			<Main>
				<div className='grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 content-stretch gap-2'>
					<Suspense fallback={<Loading message='Loading Products...' />}>
						{uiProducts.length > 0 ? (
							uiProducts.map((product) => (
								<ProductCard
									product={product}
									categoryName={categorySlug} // Use slug for links
									key={product.ProductID}
								/>
							))
						) : (
							<div className="col-span-full text-center py-10 text-white">
								<p>No products found in this category.</p>
							</div>
						)}
					</Suspense>
				</div>
			</Main>
		</>
	);
```

```
}
</file>

<file path="client/src/app/menu/page.tsx">
import { Suspense } from 'react';
import ImageGallery from '@/app/gallery/components/ImageGallery';
import Loading from '@/components/ui/Loading';
// import PromotionsBanner from '@/app/menu/components/PromotionsBanner';
import MenuSections from '@/app/menu/components/MenuSections';
import Main from '@/components/ui/layout/Main';
import { CgMenuCheese } from 'react-icons/cg';
import SearchBar from './components/SearchBar';
import { createClient } from '@/lib/supabase/server';
import { Database } from '@/lib/types/database.types';

export default async function HomePage({
      searchParams,
}: {
      searchParams: Promise<{ search?: string }>;
}) {
      const params = await searchParams;
      const search = params.search || '';
      const supabase = await createClient();

      const [productsResult, categoriesResult] = await Promise.all([
            supabase
                  .from('products')
                  .select('*')
                  .order('category_name', { ascending: true }),
            supabase
                  .from('products_category')
                  .select('*')
                  .order('id', { ascending: true })
      ]);

      const { data: productsData, error: productsError } = productsResult;
      const { data: categoriesData, error: categoriesError } = categoriesResult;

      if (productsError) {
            console.error('Error fetching products:', productsError);
      }

      if (categoriesError) {
            console.error('Error fetching categories:', categoriesError);
      }

      const allProducts: Database['public']['Tables']['products']['Row'][] =
productsData || [];
      const allCategoriesData: Database['public']['Tables']['products_category']
['Row'][] = categoriesData || [];

      // Filter out hidden categories
      const visibleCategories = allCategoriesData.filter(cat => !cat.is_hidden);
      const visibleCategoryNames = new Set(visibleCategories.map(c =>
c.category_name));

      // Filter products based on search
      const filteredProducts = search
            ? allProducts.filter((p) =>
                  p.name.toLowerCase().includes(search.toLowerCase()) ||
                  p.category_name?.toLowerCase().includes(search.toLowerCase())
            )
            : allProducts;
```

```
      // Group products by category
      const groupedProducts: ProductsType = [];
      const categoriesMap = new Map<string, ProductType[]>();

      filteredProducts.forEach((product) => {
            const category = product.category_name || 'Other';

            // Only include products in visible categories (or 'Other' if we
want to keep it, but user asked to use table)
            // Let's strictly follow the visible categories from DB for now,
unless it's a search result that might be relevant?
            // Actually, if a category is hidden, its products should probably
be hidden too in this context.
            if (!visibleCategoryNames.has(category) && category !== 'Other') {
                  // If strict mode is desired: return;
                  // But for now, let's allow 'Other' or maybe just check if it
exists in the map later.
            }

            if (!categoriesMap.has(category)) {
                  categoriesMap.set(category, []);
            }

            // Map Supabase product to UI ProductType
            const uiProduct: ProductType = {
                  ProductID: product.id,
                  Name: product.name,
                  Description: [product.description || ''],
                  Price: product.price || 0,
                  Image: product.image_url || '/Menus/placeholder.png',
                  badge: product.badge || undefined,
                  rating: product.likes ? 5 : undefined,
            };

            categoriesMap.get(category)?.push(uiProduct);
      });

      // When searching, only include categories that:
      // 1. Have matching products, OR
      // 2. Have a category name that matches the search term
      const searchLower = search.toLowerCase();
      const matchingCategoryNames = search
            ? visibleCategories
                  .filter(cat =>
cat.category_name.toLowerCase().includes(searchLower))
                  .map(cat => cat.category_name)
            : [];

      // Iterate over VISIBLE categories from DB to maintain order and metadata
      visibleCategories.forEach((category) => {
            const categoryName = category.category_name;
            const products = categoriesMap.get(categoryName) || [];

            // When searching, only show categories that have matching products
or match by name
            if (search && products.length === 0 && !
matchingCategoryNames.includes(categoryName)) {
                  return;
            }

            // If not searching, and no products, maybe skip?
            // Usually menus show empty categories if they exist, or hide them.
            // Let's hide empty categories if not searching, to be clean, unless
user wants them.
```

```
            // But the original code showed them if they were in the map.
            // Let's show if it has products OR if we are searching and it
matched.
            if (!search && products.length === 0) {
                  return;
            }

            groupedProducts.push({
                  id: category.id,
                  Name: categoryName,
                  Image: category.image || '/Menus/placeholder.png',
                  Description: category.description || '',
                  Products: products,
            });
      });

      // Handle 'Other' category if it has products and isn't in DB
      if (categoriesMap.has('Other')) {
            const otherProducts = categoriesMap.get('Other') || [];
            if (otherProducts.length > 0) {
                  groupedProducts.push({
                        id: 999,
                        Name: 'Other',
                        Image: '/Menus/placeholder.png',
                        Description: 'Other items',
                        Products: otherProducts,
                  });
            }
      }

      // Create a map of product ID to category slug for search results
      const productCategoryMap = new Map<string, string>();
      if (search) {
            filteredProducts.forEach((product) => {
                  const category = product.category_name || 'Other';
                  // Convert category name to slug (e.g., "Kota Meals" -> "kota-
meals")
                  const categorySlug = category.toLowerCase().replace(/\s+/g,
'-');
                  productCategoryMap.set(product.id, categorySlug);
            });
      }

      return (
            <>
                  <Suspense fallback={<div className="h-12" />}>
                        <SearchBar />
                  </Suspense>
                  <Main
                        tittle='Menu'
                        Icon={CgMenuCheese}
                        heading='Every Bite Matters'>
                        <Suspense fallback={<Loading message='Loading Menu
Sections...' />}>
                              <MenuSections
                                    categories={groupedProducts}
                                    matchingProducts={search ?
filteredProducts.map(p => ({
                                          ProductID: p.id,
                                          Name: p.name,
                                          Description: [p.description || ''],
                                          Price: p.price || 0,
                                          Image: p.image_url ||
'/Menus/placeholder.png',
```

```
                                        badge: p.badge || undefined,
                                        rating: undefined
                                    })) : []}
                                    productCategoryMap={search ?
productCategoryMap : undefined}
                                    isSearching={!!search}
                                />
                        </Suspense>
                        <Suspense fallback={<Loading message='Loading Image
Gallery Sections...' />}>
                                <ImageGallery />
                        </Suspense>
                  </Main>
                  {/* <PromotionsBanner /> */}
            </>
      );
}
</file>

<file path="client/src/app/orders/[State]/page.tsx">
import { createClient } from '@/lib/supabase/server';
import { redirect } from 'next/navigation';

import Main from '@/components/ui/layout/Main';
import Section from '@/components/ui/layout/Section';
import { Order } from '@/lib/supabase/orders/orders';
import OrderItem from '@/app/orders/components/OrderItem';

interface PageProps {
      params: Promise<{
            State: string;
      }>;
}

const StateOrderPage = async ({ params }: PageProps) => {
      const { State } = await params;
      const supabase = await createClient();
      const { data: { user }, error } = await supabase.auth.getUser();

      if (error || !user?.id) {
            return redirect('/login');
      }

      // Fetch orders filtered by status
      const { data: ordersData, error: ordersError } = await supabase
            .from('orders')
            .select('*')
            .eq('user_id', user.id)
            .eq('status', State.toLowerCase()) // Ensure status matches DB enum
            .order('created_at', { ascending: false });

      if (ordersError) {
            console.error('Error fetching orders:', ordersError);
      }

      const orders = (ordersData as Order[]) || [];

      if (orders.length === 0) {
            return (
                  <main className='p-4 min-h-[50vh] flex flex-col items-center
justify-center'>
                        <div className='bg-white/5 p-8 rounded-2xl border
border-white/10 text-center max-w-md'>
                              <h2 className='text-2xl font-bold text-white mb-2
```

```
capitalize'>No {State} Orders</h2>
                              <p className='text-yellow-400'>
                                    You don't have any orders with the status
"{State}" at the moment.
                              </p>
                        </div>
                  </main>
            );
      }

      return (
            <Main className='p-4 md:p-6 space-y-4'>
                  <div className='mb-6'>
                        <h2 className='text-2xl font-bold text-white mb-2
capitalize'>{State} Orders</h2>
                        <p className='text-yellow-400'>Viewing all your {State}
orders</p>
                  </div>

                  <div className='space-y-4'>
                        {orders.map((order) => (
                              <OrderItem
                                    key={order.id}
                                    order={order}
                              />
                        ))}
                  </div>
            </Main>
      );
};

export default StateOrderPage;
</file>

<file path="client/src/app/profile/components/PersonalInformation.tsx">
'use client';

import Icon from '@/components/ui/Icon';
import { FaUserCircle } from 'react-icons/fa';
import UserAddress from '../../../lib/forms/UserAddressForm';
import AppLink from '@/components/ui/Link';
import Section from '@/components/ui/layout/Section';

interface PersonalInformationProps {
      user: UserProfile;
}

const InfoRow: React.FC<{ label: string; value: string | null | undefined }> =
({
      label,
      value,
}) => (
      <div className='py-3 sm:grid sm:grid-cols-3 sm:gap-4'>
            <dt className='text-sm font-medium text-white/70'>{label}</dt>
            <dd className='mt-1 text-sm text-white sm:mt-0 sm:col-span-2'>{value
|| 'Not set'}</dd>
      </div>
);

const PersonalInformation: React.FC<PersonalInformationProps> = ({ user }) => {
      return (
            <Section
                  Icon={FaUserCircle}
                  tittle='Personal Information'>
```

```
                  <article className='mt-6 border-t border-white/20'>
                        <dl className='divide-y divide-white/20'>
                              <InfoRow
                                    label='Full Name'
                                    value={user.displayName || user.email ||
'Not set'}
                              />
                              <InfoRow
                                    label='Email address'
                                    value={user.email}
                              />
                              <InfoRow
                                    label='Phone number'
                                    value={user.phoneNumber || 'Not set'}
                              />
                              <InfoRow
                                    label='Role'
                                    value={user.role || 'User'}
                              />
                              <InfoRow
                                    label='Address'
                                    value={user.address}
                              />
                              <InfoRow
                                    label='City'
                                    value={user.city}
                              />
                              <InfoRow
                                    label='State'
                                    value={user.state}
                              />
                              <InfoRow
                                    label='Zip Code'
                                    value={user.zipCode}
                              />
                              <InfoRow
                                    label='Country'
                                    value={user.country}
                              />
                        </dl>
                  </article>
                  <div className='mt-6'>
                        <AppLink
                              href='/profile/edit'
                              className='underline text-yellow-500 hover:text-
yellow-400'>
                              Edit Profile
                        </AppLink>
                  </div>
            </Section>
      );
};

export default PersonalInformation;
</file>

<file path="client/src/components/Navbar/MobileMenu.tsx">
'use client';
import AppLink from '../ui/Link';
import { HiHome, HiInformationCircle, HiEnvelope, HiPhoto, HiCamera,
HiDocumentText, HiShoppingBag } from 'react-icons/hi2';
import { usePathname } from 'next/navigation';

const publicPaths = [
```

```
      { path: '/', icon: HiHome, label: 'Home' },
      { path: '/about', icon: HiInformationCircle, label: 'About' },
      { path: '/gallery', icon: HiPhoto, label: 'Gallery' },
      { path: '/contact', icon: HiEnvelope, label: 'Contact' },
      { path: '/photo', icon: HiCamera, label: '360 Booth' },
      { path: '/menu', icon: HiShoppingBag, label: 'Menu' },
];

const MobileMenu: React.FC<{
      setMenubar: (path: 'mobile' | 'profile') => void;
}> = ({ setMenubar }) => {
      const pathname = usePathname();

      return (
            <nav className='md:hidden absolute top-full left-0 right-0 bg-
gradient-to-b from-yellow-900/98 to-yellow-800/98 backdrop-blur-xl border-b
border-amber-400/10 shadow-2xl transition-all duration-300 z-40 max-h-screen
overflow-y-auto'>
                  <ul className='flex flex-col p-4 gap-2'>
                        {publicPaths.map(({ path, icon: Icon, label }) => {
                              const isActive = pathname === path;
                              return (
                                    <li key={path}>
                                          <AppLink
                                                href={path}
                                                onClick={() =>
setMenubar('mobile')}
                                                className={`flex items-center
gap-3 px-4 py-3 rounded-lg font-medium transition-all duration-200 w-full $
{isActive
                                                      ? 'bg-gradient-to-r from-
amber-600/20 to-yellow-600/20 border border-amber-400/30 text-amber-400 shadow-
md'
                                                      : 'text-yellow-300
hover:text-white hover:bg-white/5 border border-transparent
hover:border-white/10'
                                                      }`}>
                                                <Icon className={`text-xl $
{isActive ? 'text-amber-400' : ''}`} />
                                                <span>{label}</span>
                                          </AppLink>
                                    </li>
                              );
                        })}
                  </ul>
            </nav>
      );
};

export default MobileMenu;
</file>

<file path="client/supabase_migrations/create_tables.sql">
-- WARNING: This schema is for context only and is not meant to be run.
-- Table order and constraints may not be valid for execution.

CREATE TABLE public.admins (
  user_id uuid NOT NULL,
  created_at timestamp with time zone DEFAULT now(),
  CONSTRAINT admins_pkey PRIMARY KEY (user_id)
);
CREATE TABLE public.comments (
  id uuid NOT NULL DEFAULT gen_random_uuid(),
  product_id uuid,
```

```
  user_id uuid,
  user_name text,
  body text NOT NULL,
  created_at timestamp with time zone DEFAULT now(),
  CONSTRAINT comments_pkey PRIMARY KEY (id),
  CONSTRAINT comments_product_id_fkey FOREIGN KEY (product_id) REFERENCES
public.products(id),
  CONSTRAINT fk_comments_user_id_profiles FOREIGN KEY (user_id) REFERENCES
public.profiles(id)
);
CREATE TABLE public.contact (
  id uuid NOT NULL DEFAULT gen_random_uuid(),
  name text NOT NULL,
  email text NOT NULL,
  phone text,
  message text NOT NULL,
  created_at timestamp with time zone DEFAULT now(),
  CONSTRAINT contact_pkey PRIMARY KEY (id)
);
CREATE TABLE public.featured_items (
  id uuid NOT NULL DEFAULT gen_random_uuid(),
  name text NOT NULL,
  description text NOT NULL,
  image_url text,
  likes ARRAY DEFAULT '{}'::uuid[],
  comments jsonb DEFAULT '[]'::jsonb,
  created_at timestamp with time zone DEFAULT now(),
  CONSTRAINT featured_items_pkey PRIMARY KEY (id)
);
CREATE TABLE public.orders (
  id uuid NOT NULL DEFAULT gen_random_uuid(),
  user_id uuid,
  items jsonb NOT NULL,
  total_price numeric NOT NULL,
  total_quantity integer NOT NULL,
  status text DEFAULT 'pending'::text,
  created_at timestamp with time zone DEFAULT now(),
  updated_at timestamp with time zone DEFAULT now(),
  CONSTRAINT orders_pkey PRIMARY KEY (id),
  CONSTRAINT fk_orders_user_id_profiles FOREIGN KEY (user_id) REFERENCES
public.profiles(id)
);
CREATE TABLE public.photo_boot_bookings (
  id uuid NOT NULL DEFAULT gen_random_uuid(),
  name text NOT NULL,
  email text NOT NULL,
  phone text,
  date date NOT NULL,
  time time without time zone NOT NULL,
  package text NOT NULL,
  people integer NOT NULL,
  message text,
  created_at timestamp with time zone DEFAULT now(),
  CONSTRAINT photo_boot_bookings_pkey PRIMARY KEY (id)
);
CREATE TABLE public.products (
  id uuid NOT NULL DEFAULT gen_random_uuid(),
  name text NOT NULL,
  slug text UNIQUE,
  description text,
  price numeric DEFAULT 0,
  image_url text,
  stock integer DEFAULT 0,
  created_at timestamp with time zone DEFAULT now(),
```

```
  badge text,
  likes ARRAY DEFAULT '{}'::uuid[],
  category_name text,
  is_hidden boolean NOT NULL DEFAULT false,
  CONSTRAINT products_pkey PRIMARY KEY (id),
  CONSTRAINT products_category_name_fkey FOREIGN KEY (category_name) REFERENCES
public.products_category(category_name)
);
CREATE TABLE public.products_category (
  id uuid NOT NULL DEFAULT gen_random_uuid(),
  category_name text NOT NULL UNIQUE,
  image text,
  description text,
  created_at timestamp with time zone DEFAULT now(),
  is_hidden boolean NOT NULL DEFAULT false,
  CONSTRAINT products_category_pkey PRIMARY KEY (id)
);
CREATE TABLE public.profiles (
  id uuid NOT NULL DEFAULT gen_random_uuid(),
  email text,
  display_name text,
  phone text,
  metadata jsonb,
  created_at timestamp with time zone DEFAULT now(),
  role text DEFAULT 'customer'::text,
  uid text,
  email_verified boolean DEFAULT false,
  photo_url text,
  address text,
  city text,
  state text,
  zip_code text,
  country text,
  theme text DEFAULT 'system'::text,
  tier_status text DEFAULT 'Bronze'::text,
  referral_code text,
  preferences jsonb DEFAULT '{}'::jsonb,
  saved_payment_methods jsonb DEFAULT '[]'::jsonb,
  updated_at timestamp with time zone DEFAULT now(),
  last_login timestamp with time zone DEFAULT now(),
  CONSTRAINT profiles_pkey PRIMARY KEY (id)
);
CREATE TABLE public.testimonials (
  id uuid NOT NULL DEFAULT gen_random_uuid(),
  text text NOT NULL,
  author text NOT NULL,
  rating integer DEFAULT 5,
  likes ARRAY DEFAULT '{}'::uuid[],
  comments jsonb DEFAULT '[]'::jsonb,
  created_at timestamp with time zone DEFAULT now(),
  CONSTRAINT testimonials_pkey PRIMARY KEY (id)
);
CREATE TABLE public.user_favorites (
  user_id uuid NOT NULL,
  product_id uuid NOT NULL,
  created_at timestamp with time zone DEFAULT now(),
  CONSTRAINT user_favorites_pkey PRIMARY KEY (user_id, product_id),
  CONSTRAINT user_favorites_product_id_fkey FOREIGN KEY (product_id) REFERENCES
public.products(id),
  CONSTRAINT fk_user_favorites_user_id_profiles FOREIGN KEY (user_id) REFERENCES
public.profiles(id)
);
</file>
```

<file path="client/src/app/home-components/FeaturedItemsServices.tsx">
```
'use client';
import React, { useState, useEffect } from 'react';
import Image from 'next/image';
import { BiSolidComment, BiSolidShare } from 'react-icons/bi';
import Button from '@/components/ui/Button';
import Section from '@/components/ui/layout/Section';
import { useAuth } from '@/lib/supabase/auth/useAuth';
import { useRouter } from 'next/navigation';
import LikesButton from '@/app/menu/[Products]/components/LikesButton';
import { createClient } from '@/lib/supabase/client';

interface FeaturedItem {
      id: string;
      name: string;
      description: string;
      image_url: string | null;
      likes: string[] | null;
      comments: any[] | null;
}

const FeaturedItemsServices: React.FC = () => {
      const { user } = useAuth();
      const router = useRouter();
      const [featuredItems, setFeaturedItems] = useState<FeaturedItem[]>([]);

      useEffect(() => {
            const fetchItems = async () => {
                  const supabase = createClient();
                  const { data, error } = await supabase
                        .from('featured_items')
                        .select('*')
                        .order('created_at', { ascending: false });

                  if (error) {
                        console.error('Error fetching featured items:', error);
                  } else {
                        setFeaturedItems(data || []);
                  }
            };
            fetchItems();
      }, []);

      const handleComment = (itemId: string) => {
            if (!user) {
                  router.push('/login');
                  return;
            }
            alert('Comment functionality - would open a modal or navigate to
comments');
      };

      const handleShare = (itemId: string) => {
            if (navigator.share) {
                  navigator.share({
                        title: 'PA Luxe Creation',
                        text: 'Check out this amazing item!',
                        url: window.location.href,
                  });
            } else {
                  alert('Share functionality - URL copied to clipboard');
            }
      };
```

```
      return (
            <Section tittle='Featured Items & Services'>
                  <div className='grid grid-cols-1 md:grid-cols-2 gap-2'>
                        {featuredItems.map((item) => (
                              <article
                                    key={item.id}
                                    className='overflow-hidden flex flex-col
rounded-xl border border-yellow-500/30 bg-gradient-to-br from-yellow-900/20 to-
amber-900/20 backdrop-blur-md hover:border-yellow-400/50  transition-all
duration-300 shadow-lg'>
                                    {item.image_url && (
                                          <Image
                                                src={item.image_url}
                                                alt={item.name}
                                                width={500}
                                                height={500}
                                                className='w-full h-auto'
                                          />
                                    )}
                                    <div className='p-6 flex-grow text-center'>
                                          <h3 className='text-xl font-bold text-
white'>{item.name}</h3>
                                          <p className='mt-2 text-
yellow-300'>{item.description}</p>
                                    </div>
                                    <div className='flex justify-around items-
center border-t border-white/20 pt-4 bg-white/5 -mx-6 px-6 pb-6 -mb-6 mt-auto
rounded-b-xl'>
                                          <div className="flex flex-col items-
center gap-1">
                                                <LikesButton itemId={item.id}
table="featured_items" />
                                          </div>
                                          <Button
                                                variant='icon'
                                                onClick={() =>
handleComment(item.id)}
                                                className='flex flex-col items-
center gap-1'
                                                aria-label='Comment on item'>
                                                <BiSolidComment
                                                      size={24}
                                                      className='text-white
hover:text-yellow-500 transition-colors'
                                                />
                                                <span className='text-xs text-
yellow-400'>{(item.comments as any[])?.length || 0}</span>
                                          </Button>
                                          <Button
                                                variant='icon'
                                                onClick={() =>
handleShare(item.id)}
                                                className='flex flex-col items-
center gap-1'
                                                aria-label='Share item'>
                                                <BiSolidShare
                                                      size={24}
                                                      className='text-white
hover:text-yellow-500 transition-colors'
                                                />
                                                <span className='text-xs text-
yellow-400'>Share</span>
                                          </Button>
                                    </div>
```

```
                              </article>
                        ))}
                  </div>
            </Section>
      );
};

export default FeaturedItemsServices;
</file>

<file path="client/src/app/menu/components/SearchBar.tsx">
'use client';
import React, { useState } from 'react';
import { HiSearch } from 'react-icons/hi';
import TextInput from '@/components/ui/TextInput';
import { useRouter, useSearchParams, usePathname } from 'next/navigation';

const SearchBar: React.FC = () => {
      const searchParams = useSearchParams();
      const router = useRouter();
      const pathname = usePathname();
      const [term, setTerm] = useState(searchParams.get('search') || '');

      const handleSearch = (value: string) => {
            setTerm(value);
            const params = new URLSearchParams(searchParams);
            if (value) {
                  params.set('search', value);
            } else {
                  params.delete('search');
            }
            router.replace(`${pathname}?${params.toString()}`);
      };

      return (
            <div className='bg-gradient-to-br from-yellow-900/20 to-amber-900/20
backdrop-blur-md p-3 rounded-b-xl border border-amber-400/10 sticky top-20 z-20
col-span-full mt-0 shadow-xl shadow-black/20'>
                  <div className='max-w-7xl mx-auto relative w-full'>
                        <TextInput
                              id='menu-search'
                              className='bg-yellow-900/60 border-amber-400/20
focus:border-amber-400 pl-12'
                              type='text'
                              placeholder='Search for food or categories...'
                              value={term}
                              onChange={(e) => handleSearch(e.target.value)}
                              aria-label='Search for food or categories'
                              autoComplete='off'
                              icon={
                                    <HiSearch
                                          className='text-amber-400'
                                          size={20}
                                          aria-hidden='true'
                                    />
                              }
                        />
                  </div>
            </div>
      );
};

export default SearchBar;
</file>
```

```
<file path="client/src/components/Navbar/DesktopNavbar.tsx">
'use client';
import AppLink from '@/components/ui/Link';
import Image from 'next/image';
import {
      HiHome,
      HiInformationCircle,
      HiEnvelope,
      HiPhoto,
      HiCamera,
      HiDocumentText,
      HiShoppingBag,
} from 'react-icons/hi2';
import React, { useState, useRef, useEffect } from 'react';
import Button from '@/components/ui/Button';
import { useAuth } from '@/lib/supabase/auth/useAuth';
import { usePathname } from 'next/navigation';

const publicPaths = [
      { path: '/', icon: HiHome, label: 'Home' },
      { path: '/about', icon: HiInformationCircle, label: 'About' },
      { path: '/gallery', icon: HiPhoto, label: 'Gallery' },
      { path: '/contact', icon: HiEnvelope, label: 'Contact' },
      { path: '/photo', icon: HiCamera, label: '360 Booth' },
      { path: '/menu', icon: HiShoppingBag, label: 'Menu' },
];

const DesktopNavbar: React.FC = () => {
      const { user } = useAuth();
      const pathname = usePathname();
      const [profileOpen, setProfileOpen] = useState(false);
      const profileRef = useRef<HTMLLIElement>(null);

      useEffect(() => {
            const handleClickOutside = (event: MouseEvent) => {
                  if (profileRef.current && !
profileRef.current.contains(event.target as Node)) {
                        setProfileOpen(false);
                  }
            };
            document.addEventListener('mousedown', handleClickOutside);
            return () => {
                  document.removeEventListener('mousedown', handleClickOutside);
            };
      }, []);

      return (
            <nav className='hidden md:flex gap-4 justify-between items-center
p-0 m-0 w-full'>
                  <AppLink href='/' className='shrink-0'>
                        <Image
                              src='/PA_Logo.png'
                              alt='PA Catering Logo'
                              width={64}
                              height={64}
                              className='h-16 w-auto'
                              priority
                        />
                  </AppLink>

                  <ul className='flex space-x-2 items-center'>
                        {publicPaths.map(({ path, icon: Icon, label }) => {
                              const isActive = pathname === path;
```

```
                              return (
                                    <li key={path}>
                                          <AppLink
                                                href={path}
                                                className={`flex items-center
gap-2 px-4 py-2 rounded-lg font-medium transition-all duration-200 ${isActive
                                                      ? 'bg-gradient-to-r from-
amber-600/20 to-yellow-600/20 border border-amber-400/30 text-amber-400 shadow-
md shadow-amber-500/10'
                                                      : 'text-yellow-300
hover:text-white hover:bg-white/5 border border-transparent
hover:border-white/10'
                                                      }`}>
                                                <Icon className={`text-lg $
{isActive ? 'text-amber-400' : ''}`} />
                                                <span>{label}</span>
                                          </AppLink>
                                    </li>
                              );
                        })}
                  </ul>
            </nav>
      );
};

export default DesktopNavbar;
</file>

<file path="client/src/components/Navbar/ProfileMenu.tsx">
'use client';
import AppLink from '../ui/Link';
import { HiUser, HiClipboardList, HiShoppingBag, HiLogout, HiCamera } from
'react-icons/hi';
import Button from '../ui/Button';
import { User } from '@supabase/supabase-js';
import { usePathname } from 'next/navigation';

const protectedPaths = [
      { path: '/profile', icon: HiUser, label: 'Profile' },
      { path: '/orders', icon: HiClipboardList, label: 'Orders' },

      { path: '/photo/booking', icon: HiCamera, label: '360 Booth Booking' },
];

const ProfileMenu: React.FC<{ setMenubar: (path: 'mobile' | 'profile') => void;
user: User | null }> = ({
      setMenubar,
      user,
}) => {
      const pathname = usePathname();

      if (!user) return null;

      return (
            <nav className='absolute top-full right-0 w-full sm:w-80 bg-
gradient-to-b from-yellow-900/98 to-yellow-800/98 backdrop-blur-xl border
border-amber-400/10 rounded-b-xl shadow-2xl transition-all duration-300 z-40'>
                  {/* User Info */}
                  <div className='px-4 py-3 border-b border-white/10'>
                        <p className='text-sm text-yellow-400'>Signed in as</p>
                        <p className='text-white font-semibold
truncate'>{user.email}</p>
                  </div>
```

```
                  {/* Menu Items */}
                  <ul className='flex flex-col p-2'>
                        {protectedPaths.map(({ path, icon: Icon, label }) => {
                              const isActive = pathname === path;
                              return (
                                    <li key={path}>
                                          <AppLink
                                                href={path}
                                                onClick={() =>
setMenubar('profile')}
                                                className={`flex items-center
gap-3 px-4 py-2.5 rounded-lg font-medium transition-all duration-200 w-full $
{isActive
                                                      ? 'bg-gradient-to-r from-
amber-600/20 to-yellow-600/20 border border-amber-400/30 text-amber-400'
                                                      : 'text-yellow-300
hover:text-white hover:bg-white/5 border border-transparent
hover:border-white/10'
                                                      }`}>
                                                <Icon className={`text-lg $
{isActive ? 'text-amber-400' : ''}`} />
                                                <span>{label}</span>
                                          </AppLink>
                                    </li>
                              );
                        })}
                  </ul>

                  {/* Logout Button */}
                  <div className='p-2 border-t border-white/10'>
                        <Button
                              variant='primary'
                              onClick={async () => {
                                    const { createClient } = await
import('@/lib/supabase/client');
                                    const supabase = createClient();
                                    await supabase.auth.signOut();
                                    setMenubar('profile');
                              }}
                              className='flex items-center justify-center gap-2
px-4 py-2.5 w-full bg-red-500/10 hover:bg-red-500/20 border border-red-500/30
hover:border-red-500/50 text-red-300 hover:text-red-200 rounded-lg font-semibold
transition-all duration-200'>
                              <HiLogout className='text-lg' />
                              <span>Log Out</span>
                        </Button>
                  </div>
            </nav>
      );
};

export default ProfileMenu;
</file>

<file path="client/src/components/ui/layout/Section.tsx">
import { Suspense } from 'react';
import Loading from '../Loading';

interface SectionProps extends React.HTMLProps<HTMLElement> {
      Icon?: React.ElementType;
      heading?: string;
      tittle?: string;
      className?: string;
      children: React.ReactNode;
```

```
}
export default function Section({
      children,
      Icon,
      heading,
      tittle,
      className,
      ...props
}: SectionProps) {
      return (
            <section
                  {...props}
                  className={
                        className ? className : 'bg-gradient-to-br from-
yellow-900/20 to-amber-900/20 backdrop-blur-md border border-yellow-500/30
rounded-xl p-6 hover:border-yellow-400/50 transition-all duration-300 w-full
my-4 md:my-6 lg:my-8'
                  }>
                  <header>
                        {Icon ? (
                              <div className='flex flex-col items-center text-
center'>
                                    <span className='bg-gradient-to-br from-
amber-500/20 to-yellow-500/20 p-3 rounded-lg border border-amber-400/30 w-fit
mb-4'>
                                          <Icon className='text-amber-400
text-3xl' />
                                    </span>
                                    {tittle && <h3 className='mt-4 text-shadow-
sm text-shadow-black/50'>{tittle}</h3>}
                              </div>
                        ) : (
                              tittle && <h2 className='text-shadow-sm text-
shadow-black/50'>{tittle}</h2>
                        )}
                        <p className='text-white/70 text-center mt-2 flex-
grow'>{heading}</p>
                  </header>
                  {children}
            </section>
      );
}
</file>

<file path="client/src/app/orders/page.tsx">
export const dynamic = 'force-dynamic';
import { createClient } from '@/lib/supabase/server';
import { getOrdersByUser, Order } from '@/lib/supabase/orders/orders';
import OrderItem from '@/app/orders/components/OrderItem';
import Main from '@/components/ui/layout/Main';
import { HiDocumentText, HiShoppingBag } from 'react-icons/hi2';
import { redirect } from 'next/navigation';

const OrderHistory: React.FC = async () => {
      const supabase = await createClient();
      const { data: { user }, error } = await supabase.auth.getUser();

      if (error || !user?.id) {
            return redirect('/login');
      }

      const { data: ordersData, error: ordersError } = await supabase
            .from('orders')
            .select('*')
```

```
            .eq('user_id', user.id)
            .order('created_at', { ascending: false });

      if (ordersError) {
            console.error('Error fetching orders:', ordersError);
      }

      // ... existing code ...

      // ... existing code ...

      const orders = (ordersData as Order[]) || [];

      if (orders.length === 0) {
            return (
                  <Main
                        tittle='Order History'
                        Icon={HiDocumentText}
                        className='p-4 flex items-center justify-center min-h-
[60vh]'>
                        <div className='text-center bg-gradient-to-br from-
yellow-900/90 to-yellow-800/90 backdrop-blur-xl p-12 rounded-2xl border border-
white/10 shadow-2xl max-w-md'>
                              <div className='flex flex-col items-center gap-4'>
                                    <div className='bg-gradient-to-br from-
amber-500/20 to-yellow-500/20 p-6 rounded-full border border-amber-400/30'>
                                          <HiShoppingBag className='text-6xl
text-amber-400' />
                                    </div>
                                    <h2 className='text-3xl font-bold text-
white'>
                                          No Orders Yet
                                    </h2>
                                    <p className='text-yellow-400 max-w-sm'>
                                          Your order history will appear here
once you place your first order.
                                    </p>
                              </div>
                        </div>
                  </Main>
            );
      }

      return (
            <Main
                  tittle='Order History'
                  Icon={HiDocumentText}
                  className='p-4 md:p-6 space-y-4'>
                  <div className='mb-6'>
                        <h2 className='text-2xl font-bold text-white mb-2'>Your
Orders</h2>
                        <p className='text-yellow-400'>Track and manage all your
orders in one place</p>
                  </div>

                  <div className='space-y-4'>
                        {orders.map((order) => (
                              <OrderItem
                                    key={order.id}
                                    order={order}
                              />
                        ))}
                  </div>
            </Main>
```

```
      );
};

export default OrderHistory;
</file>

<file path="client/src/lib/actions/auth.ts">
"use server";

import { createClient } from "@/lib/supabase/server";
import { redirect } from "next/navigation";
import type { User, SupabaseClient } from "@supabase/supabase-js";
import { Database } from "@/lib/types/database.types";

export type AuthActionState = {
  success: boolean;
  error?: string;
  user?: User | null;
};

export async function signInAction(
  _prevState: AuthActionState | null,
  formData: FormData
): Promise<AuthActionState> {
  const email = String(formData.get("email") ?? "").trim();
  const password = String(formData.get("password") ?? "").trim();

  if (!email || !password) {
    return {
      success: false,
      error: "Email and password are required.",
    };
  }

  let user = null;

  try {
    const supabase: SupabaseClient<Database> = await createClient();
    const { data, error } = await supabase.auth.signInWithPassword({
      email,
      password,
    });

    if (error) {
      return {
        success: false,
        error: error.message,
      };
    }

    user = data.user;
  } catch (error) {
    return {
      success: false,
      error: error instanceof Error ? error.message : "An unexpected error
occurred during sign in.",
    };
  }

  if (user) {
    redirect("/profile");
  }

  return {
```

```
    success: false,
    error: "Sign in failed. Please check your credentials.",
  };
}

export async function signUpAction(
  _prevState: AuthActionState | null,
  formData: FormData
): Promise<AuthActionState> {
  const email = String(formData.get("email") ?? "").trim();
  const password = String(formData.get("password") ?? "").trim();
  const displayName = String(formData.get("displayName") ?? "").trim();
  const phoneNumber = String(formData.get("phoneNumber") ?? "").trim() ||
undefined;

  if (!email || !password || !displayName) {
    return {
      success: false,
      error: "Email, password, and display name are required.",
    };
  }

  let user = null;

  try {
    const supabase: SupabaseClient<Database> = await createClient();

    // Sign up user
    const { data: authData, error: authError } = await supabase.auth.signUp({
      email,
      password,
      options: {
        data: {
          display_name: displayName,
          phone_number: phoneNumber || "",
        },
      },
    });

    if (authError) {
      return {
        success: false,
        error: authError.message,
      };
    }

    user = authData.user;

    // Profile creation is handled by Postgres trigger on auth.users insert
    // See: supabase_migrations/add_profile_trigger.sql
  } catch (error) {
    return {
      success: false,
      error: error instanceof Error ? error.message : "An unexpected error
occurred during registration.",
    };
  }

  if (user) {
    redirect("/confirm-email");
  }

  return {
    success: false,
```

```
    error: "Registration failed. Please try again.",
  };
}
</file>

<file path="client/src/app/profile/page.tsx">
import PersonalInformation from './components/PersonalInformation';

import { FaUserCircle } from 'react-icons/fa';
import Main from '@/components/ui/layout/Main';

import { createClient } from "@/lib/supabase/server";
import { redirect } from "next/navigation";

const CustomerProfilePage: React.FC = async () => {
      const supabase = await createClient();
      const { data: { user } } = await supabase.auth.getUser();

      if (!user) {
            redirect("/login");
      }

      const { data: profileData } = await supabase
            .from("profiles")
            .select("*")
            .eq("id", user.id)
            .single();

      if (!profileData) {
            // Handle case where profile doesn't exist (shouldn't happen if auth
works)
            return <div>Profile not found</div>;
      }

      const profile = profileData as any;

      // Map DB profile (snake_case) to UserProfile (camelCase)
      const userProfile: UserProfile = {
            uid: profile.uid || profile.id,
            displayName: profile.display_name,
            email: profile.email,
            emailVerified: profile.email_verified || false,
            photoURL: profile.photo_url,
            phoneNumber: profile.phone,
            role: (profile.role as UserRole) || 'Customer',
            address: profile.address || '',
            city: profile.city || '',
            state: profile.state || '',
            zipCode: profile.zip_code || '',
            country: profile.country || '',
            theme: (profile.theme as 'system' | 'light' | 'dark') || 'system',
            tierStatus: profile.tier_status || 'Bronze',
            referralCode: profile.referral_code || '',
            preferences: (profile.preferences as any) || {
                  dietaryRestrictions: [],
                  favoriteItems: [],
                  preferyellowPaymentMethod: 'credit_card',
                  communicationPreferences: { email: true, sms: false,
promotions: true }
            },
            savedPaymentMethods: (profile.saved_payment_methods as any) || [],
            createdAt: new Date(profile.created_at),
            updatedAt: profile.updated_at ? new Date(profile.updated_at) : new
Date(),
```

```
            lastLogin: profile.last_login ? new Date(profile.last_login) :
undefined,
      };

      return (
            <Main
                  tittle='Profile'
                  Icon={FaUserCircle}
                  heading='Manage your personal information, preferences, and
account security.'>

                  <PersonalInformation user={userProfile} />

            </Main>
      );
};

export default CustomerProfilePage;
</file>

</files>
```