BestChoose

BestChoose to aplikacja webowa (Next.js App Router) dla platformy medycznej z uwierzytelnianiem w Supabase oraz panelami dla ról: **admin / doctor / patient**.

Dokumentacja

- **Repozytorium**: `https://github.com/kyrylokap/BestChoose`
- **Dokumentacja PDF**: docs/BestChoose_Dokumentacja.pdf

Autorzy

Projekt został wykonany przez 2 osoby:

- **Kyrylo Kapinos**
- **Vasyl Ishchuk**

Opis aplikacji i funkcjonalności

- **Uwierzytelnianie**: logowanie/rejestracja/reset hasła oparte o Supabase Auth
- **Autoryzacja i routing per rola**: public/protected routes + przekierowania do właściwego dashboardu
- **Panel admina**: zarządzanie lekarzami (lista, wyszukiwanie z debounce, paginacja) + strona statystyk z wykresami
- **Panel lekarza**: podgląd nadchodzących wizyt (harmonogram) i przejście do szczegółów wizyty
- **Portal pacjenta**: szybki start wywiadu z AI + lista nadchodzących wizyt
- **UX**: spójne loadery podczas sprawdzania sesji/przekierowań + powiadomienia toast

Wymagania (requirements)

- **Node.js**: v20+ (v18 może działać, ale zalecane jest v20+)
- **Package manager**: yarn (zalecany) / npm / pnpm / bun
- **Supabase**: projekt Supabase + klucze (patrz sekcja env)
- **Opcjonalnie**: Supabase CLI (jeśli chcesz generować typy przez `yarn generate`)

Stack technologiczny

- **Framework**: Next.js 16 (App Router)
- **UI**: React 19, Tailwind CSS v4, Headless UI, lucide-react
- **Formy i walidacja**: react-hook-form + zod
- **Wykresy**: victory
- **Auth/DB**: Supabase (`@supabase/supabase-js`, `@supabase/ssr`)
- **Notyfikacje**: sonner

Konfiguracja środowiska (env)

Utwórz plik `.env.local` (zalecane) i dodaj:

```
NEXT_PUBLIC_SUPABASE_URL=
NEXT_PUBLIC_SUPABASE_ANON_KEY=
NEXT_PUBLIC_SERVICE_ROLE_KEY=
```

Ważne (bezpieczeństwo)

Zmienne `NEXT_PUBLIC_*` są wystawiane do przeglądarki w Next.js. **Service role key nie może być publiczny**.

Ten projekt aktualnie czyta `NEXT_PUBLIC_SERVICE_ROLE_KEY` (zob. `src/api/supabaseAdmin.ts`). Traktuj go jako **super wrażliwy**:

- **Nigdy go nie commituj**
- **Nigdy nie używaj go w kodzie client-side**
- Docelowo przenieś go do zmiennej **server-only** (bez prefixu `NEXT_PUBLIC_`) i używaj tylko w API/route handlers/actions po stronie serwera

Jak uruchomić aplikację (krok po kroku)

1) Instalacja zależności

```
yarn
```

2) Konfiguracja env

- Utwórz `.env.local`
- Uzupełnij wymagane zmienne Supabase (powyżej)

3) Start w trybie developerskim

```
yarn dev
```

Otwórz `http://localhost:3000`.

Skrypty (npm/yarn scripts)

- **Dev**:

```
yarn dev
```

- **Lint**:

```
yarn lint
```

- **Build (produkcja)**:

```
yarn build
```

- **Start (produkcja)** (po buildzie):

```
yarn start
```

- **Generowanie typów Supabase** (opcjonalnie; wymaga Supabase CLI + dostępu do projektu):

```
yarn generate
```

Struktura projektu

Najważniejsze katalogi i pliki:

- **Routing / widoki (App Router)**: `src/app/`
  - dashboardy: `src/app/(dashboard)/`
- **Komponenty**: `src/components/`
  - dashboardy: `src/components/dashboards/`
  - route guards / sesja: `src/components/hoc/`
  - elementy współdzielone: `src/components/shared/`
- **Hooki**: `src/hooks/`
- **Supabase clienty**: `src/api/supabase.ts`, `src/api/supabaseAdmin.ts`
- **Dane statyczne**: `src/data/`
- **Typy**: `src/types/`

Jak działa logika dostępu (w skrócie)

- **Public routes** (np. login/register): przekierowują zalogowanych użytkowników do odpowiedniego dashboardu.
- **Protected routes** (dashboardy): wymagają sesji i roli; w trakcie sprawdzania sesji pokazują loader, aby uniknąć "flashowania" nieautoryzowanych stron.

Troubleshooting

- **Pusta strona lub pętla przekierowań**: sprawdź env i czy URL/klucze Supabase są z właściwego projektu.
- **Wykresy nic nie pokazują**: upewnij się, że w bazie są dane (np. appointments/reports) i aplikacja ma uprawnienia do odczytu.
- **`yarn generate` nie działa**: zainstaluj i zaloguj Supabase CLI, i sprawdź czy `project-id` w skrypcie pasuje do Twojego projektu.