```
'use client';

import { Github, LogIn } from 'lucide-react';
import { FormEvent, useMemo, useRef, useState } from 'react';
import { usePathname, useRouter } from 'next/navigation';
import Logo from '@/shared/components/icons/Logo';
import { cn } from '@/shared/utils/cn';
import { User } from '@supabase/supabase-js';
import useToggle from '@/shared/hooks/toggle';
import { SignOut } from '@/shared/utils/auth-helpers/server';
import { useOnClickOutside } from '@/shared/hooks/onclick-outside';
import { handleRequest } from '@/shared/utils/auth-helpers/client';
import { NAV_LINKS } from '../const';
import NavItem from '../item';
import NavbarSearch from './components/search';
import { trpc } from '@/shared/utils/trpc/trpc';
import Avatar from '@/shared/components/avatar';
import { NAVIGATION, SWARMS_GITHUB } from '@/shared/constants/links';

export default function PlatformNavBar({ user }: { user: User | null }) {
  const dropdownRef = useRef(null);
  const path = usePathname();
  const router = useRouter();
  const { isOn, setOn, setOff } = useToggle();
  const username = trpc.main.getUser.useQuery().data?.username;
  const profileName = username ? username : user?.user_metadata?.email;
```

```
const [isLoading, setIsLoading] = useState(false);


const isSwarmsPath = path === '/swarms';


const FILTERED_NAV_LINKS = useMemo(
  () =>
    !isSwarmsPath
      ? NAV_LINKS.external
      : NAV_LINKS.external
          ?.filter(
            (item) =>
              item.link !== NAVIGATION.PRICING &&
              item.link !== NAVIGATION.GET_DEMO,
          )
          .concat([
            {
              icon: <Github />,
              title: 'Github',
              link: SWARMS_GITHUB,
            },
          ]),
  [isSwarmsPath],
);


async function handleSignOut(e: FormEvent<HTMLFormElement>) {
  setIsLoading(true);
```

```
    try {
      await handleRequest(e, SignOut, router);
    } catch (error) {
      console.error(error);
    } finally {
      setIsLoading(false);
    }
  }

  useOnClickOutside(dropdownRef, setOff);
  return (
      <header  className="fixed  max-sm:flex  max-sm:items-center  w-full  top-0  backdrop-blur-sm
bg-black shadow-md z-40 transition-all duration-150 px-4 py-2 h-16 md:h-20">
    <nav className="flex items-center justify-between max-sm:w-full relative">
      <div className="flex items-center w-4/5 sm:w-1/2 xl:w-1/3">
        <div className="flex items-center w-[40px] h-[40px] min-w-[40px] max-lg:hidden mr-4">
          <Logo />
        </div>

        <NavbarSearch />
      </div>
      <div className="flex items-center space-x-4">
        <ul className="p-0 hidden items-center sm:flex">
          {FILTERED_NAV_LINKS?.map((item) => (
            <li key={item.title}>
              <NavItem
```

```
            {...item}
            className={cn(
              'text-white p-2 py-3 my-1 hover:text-primary',
              item.link === path && 'text-primary',
            )}
            showTitle
          >
            {item.title}
          </NavItem>
        </li>
      ))}
    </ul>
    <div
      className="relative ml-5 cursor-pointer max-sm:mt-1"
      onClick={setOn}
    >
      <Avatar user={user as User} profileName={profileName} />
      <ul
        ref={dropdownRef}
        className={cn(
          "absolute right-0 mt-4 w-72 group-hover:block z-10 p-0 transition duration-150 invisible bg-black/85 text-white border border-secondary bg-opacity-75 rounded-md shadow-lg before:content-[''] before:absolute before:translate-x-1/4 before:block before:border-[10px] before:border-solid before:border-white before:border-t-[transparent] before:border-r-[transparent] before:border-b-secondary before:border-l-[transparent] before:right-3 before:-top-5",
          isOn && 'translate-x-0 visible',
```

```
          )}
        >
          <li
            className={cn(
              'p-4 text-sm rounded-t-md border-b-slate-800 border-b flex justify-between items-center',
            )}
          >
            <NavItem title={profileName} />
            <span
              title="Active user"
                  className="inline-flex shrink-0 items-center justify-center whitespace-nowrap text-sm border border-input bg-secondary text-foreground font-medium shadow-sm hover:bg-accent hover:text-accent-foreground h-8 py-2 gap-[6px] rounded-full px-2"
            >
              {profileName?.charAt(0)?.toUpperCase()}_swarms+
            </span>
          </li>
          {NAV_LINKS.account?.map((item, index, array) => {
            const isLast = index === array.length - 1;
            return (
              <li
                key={item.title}
                className={cn(
                  'text-sm hover:bg-destructive hover:text-white ',
                  item.link === path && 'bg-primary text-white',
                  isLast && 'rounded-b-md border-t-slate-800 border-t',
```

```
  )}
>
  {!item.link ? (
    user?.role === 'authenticated' ? (
      <NavItem
        {...item}
        as="form"
        isIcon
        className="w-full p-4"
        onSubmit={handleSignOut}
      >
        <input
          type="hidden"
          name="pathName"
          value={usePathname()?.toString()}
        />
        <button
          type="submit"
          className="flex items-center w-full"
        >
          {isLoading ? 'Signin out...' : 'Sign out'}
        </button>
      </NavItem>
    ) : (
      <NavItem
        link="/signin"
```

```
                    title="Sign in"
                    className="p-4"
                    isIcon
                    icon={<LogIn size={20} />}
                  />
                )
              ) : (
                <NavItem {...item} isIcon showTitle />
              )}
            </li>
          );
        })}
      </ul>
    </div>
  </div>
</nav>
</header>
);
}
```