```
'use client';

import {
  Toast,
  ToastClose,
  ToastDescription,
  ToastProvider,
  ToastTitle,
  ToastViewport,
} from '@/shared/components/ui/Toasts/toast';
import { useToast } from '@/shared/components/ui/Toasts/use-toast';
import { usePathname, useRouter, useSearchParams } from 'next/navigation';
import { useEffect } from 'react';

export function Toaster() {
  const { toast, toasts } = useToast();
  const searchParams = useSearchParams();
  const pathname = usePathname();
  const router = useRouter();

  useEffect(() => {
    const status = searchParams?.get('status');
    const status_description = searchParams?.get('status_description');
    const error = searchParams?.get('error');
    const error_description = searchParams?.get('error_description');
    if (error || status) {
```

```
    toast({
      title: error
        ? (error ?? 'Hmm... Something went wrong.')
        : (status ?? 'Alright!'),
      description: error ? error_description : status_description,
      variant: error ? 'destructive' : undefined,
    });
    // Clear any 'error', 'status', 'status_description', and 'error_description' search params
    // so that the toast doesn't show up again on refresh, but leave any other search params
    // intact.
    const newSearchParams = new URLSearchParams(searchParams?.toString());
    const paramsToRemove = [
      'error',
      'status',
      'status_description',
      'error_description',
    ];
    paramsToRemove.forEach((param) => newSearchParams.delete(param));
    const redirectPath = `${pathname}?${newSearchParams.toString()}`;
    router.replace(redirectPath, { scroll: false });
  }
}, [searchParams]);

return (
  <ToastProvider>
    {toasts.map(function ({ id, title, description, action, ...props }) {
```

```
        return (
          <Toast key={id} {...props}>
            <div className="grid gap-1">
              {title && <ToastTitle>{title}</ToastTitle>}
              {description && (
                <ToastDescription>{description}</ToastDescription>
              )}
            </div>
            {action}
            <ToastClose />
          </Toast>
        );
      })}
      <ToastViewport />
    </ToastProvider>
  );
}
```