```
import React, {

  Dispatch,

  FormEvent,

  memo,

  SetStateAction,

  useRef,

  useState,

} from 'react';

import { Ellipsis } from 'lucide-react';

import {

  Dialog,

  DialogContent,

  DialogFooter,

  DialogHeader,

  DialogTitle,

  DialogTrigger,

} from '@/shared/components/ui/dialog';

import { Button } from '@/shared/components/ui/Button';

import Input from '@/shared/components/ui/Input';

import { cn } from '@/shared/utils/cn';

import useToggle from '@/shared/hooks/toggle';

import { useOnClickOutside } from '@/shared/hooks/onclick-outside';

import { getTruncatedString } from '@/shared/utils/helpers';

import { Role } from '../../../types';

import { useOrganizationStore } from '@/shared/stores/organization';

import LoadingSpinner from '@/shared/components/loading-spinner';
```

```
interface OrganizationListItemProps {
  name: string;
  role: Role;
  openDialog?: boolean;
  setOpenDialog?: Dispatch<SetStateAction<boolean>>;
  handleCurrentOrgId?: () => void;
  updateOrganization?: (event: FormEvent<HTMLFormElement>) => void;
}

function OrganizationListItem({
  name,
  role,
  openDialog,
  setOpenDialog,
  updateOrganization,
  handleCurrentOrgId,
}: OrganizationListItemProps) {
  const isLoading = useOrganizationStore((state) => state.isLoading);

  const formRef = useRef<HTMLFormElement>(null);
  const { isOn, setOff, setOn } = useToggle();
  const popupRef = useRef(null);
  const [organizationName, setOrganizationName] = useState('');
  const orgName = getTruncatedString(name, 20);
  useOnClickOutside(popupRef, setOff);


  return (
    <div
      onClick={handleCurrentOrgId}
      className="flex justify-between border rounded-md p-2 sm:p-4 text-card-foreground hover:opacity-90 w-full transition cursor-pointer shadow bg-white text-black dark:bg-black dark:text-white"
    >
      <div className="flex items-center gap-2">
        <span className="h-7 w-7 sm:w-10 sm:h-10 text-sm sm:text-base flex justify-center items-center bg-secondary text-foreground rounded-full uppercase">
          {name.charAt(0)}
        </span>
        <p className="text-xs sm:text-base">{orgName}</p>
      </div>
      <div className="flex items-center gap-4 sm:justify-between sm:max-w-48 sm:w-full">
        <p className="capitalize text-xs sm:text-base">{role}</p>
        <div className="relative">
          <Button
            aria-label="Options"
            className={cn(
              'gap-2 py-0 h-8 px-2 sm:px-4 sm:h-9',
              isOn && 'bg-accent text-foreground',
            )}
            variant="ghost"
            onClick={setOn}
          >
            <Ellipsis />
          </Button>

          <div
            ref={popupRef}
            className={cn(
              'absolute list-none border dark:border-white/[0.2] bg-secondary max-w-28 w-full flex flex-col
  items-center rounded-md bottom-8 left-0 transition-all invisible',
              isOn && 'visible',
            )}
          >
            {role === 'owner' && (
              <Dialog open={openDialog} onOpenChange={setOpenDialog}>
                <DialogTrigger asChild>
                  <div
                    onClick={setOff}
                    className="hover:text-secondary hover:bg-foreground text-foreground capitalize w-full
  py-2 text-center"
                  >
                    Edit
                  </div>
                </DialogTrigger>
                <DialogContent className="max-w-[320px] sm:max-w-[425px]">
                  <DialogHeader>
```

```
  <DialogTitle>Update organization name</DialogTitle>
</DialogHeader>
<form
  ref={formRef}
  onSubmit={updateOrganization}
  className="mt-2"
>
  <label htmlFor="name" className="text-right">
    Name
  </label>
  <Input
    id="name"
    name="name"
    aria-label="Name"
    value={organizationName}
    className="my-2 w-full"
    onChange={(value) => {
      setOrganizationName(value);
    }}
  />
  <DialogFooter className="mt-3 sm:justify-center">
    <Button
      type="submit"
      className="w-2/4"
      aria-label="Edit organization"
    >
```

```
                        {isLoading ? <LoadingSpinner /> : 'Edit'}
                      </Button>
                    </DialogFooter>
                  </form>
                </DialogContent>
              </Dialog>
            )}
          </div>
        </div>
      </div>
    </div>
  );
}

export default memo(OrganizationListItem);
```