```
import React, { memo, useState } from 'react';
import { ShieldX } from 'lucide-react';
import { Button } from '@/shared/components/ui/Button';
import InviteModal from './team/components/invite-modal';
import LoadingSpinner from '@/shared/components/loading-spinner';
import { useToast } from '@/shared/components/ui/Toasts/use-toast';
import { useOrganizationStore } from '@/shared/stores/organization';
import { PendingInvitesProps, UserOrganizationsProps } from '../types';
import ModalPrompt from './prompt';
import { isEmpty } from '@/shared/utils/helpers';
import { useQueryMutation } from '../hooks/organizations';

function PendingInvites({
  currentOrganization,
}: {
  currentOrganization: UserOrganizationsProps;
}) {
  const { query, mutation } = useQueryMutation();

  const toast = useToast();

  const isLoading = useOrganizationStore((state) => state.isLoading);
  const userOrgId = useOrganizationStore((state) => state.userOrgId);
  const [openDialog, setOpenDialog] = useState(false);

  const isOwnerOrManager =
```

```
  currentOrganization?.role === 'owner' ||
  currentOrganization?.role === 'manager';

// cancel pending user invite
async function handleCancelInvite(email: string) {
  useOrganizationStore.getState().setIsLoading(true);
  try {
    const response = await mutation.cancel.mutateAsync({
      email,
      organization_id: userOrgId ?? '',
    });
    if (response) {
      toast.toast({ description: `Invite has been cancelled for ${email}` });
      query?.invites?.refetch();
    }
  } catch (error) {
    if ((error as any)?.message) {
      console.error(error);
      toast.toast({ description: (error as any)?.message });
    }
  } finally {
    useOrganizationStore.getState().setIsLoading(false);
  }
}

function renderItem({ id, email }: PendingInvitesProps) {
```

```
    return (
      <div
        key={id}
              className="flex justify-between border rounded-md px-4 py-8 text-card-foreground hover:opacity-80 w-full cursor-pointer mb-4"
      >
        <div className="flex items-center gap-2">
                    <span className="w-10 h-10 flex justify-center items-center bg-foreground dark:bg-secondary text-white rounded-full uppercase">
            {email?.charAt(0)}
          </span>
          <p>{email}</p>
        </div>
        <div className="flex items-center gap-4">
          <ModalPrompt
            content={`Do you wish to cancel invite for ${email}?`}
            isLoading={isLoading}
            handleClick={() => handleCancelInvite(email as string)}
            openDialog={openDialog}
            setOpenDialog={setOpenDialog}
          >
            <Button
              variant="destructive"
              aria-label="Cancel invite"
              className="flex gap-2"
            >
```

```
            Cancel <ShieldX size={20} />
          </Button>
        </ModalPrompt>
      </div>
    </div>
  );
}

if (isOwnerOrManager)
  return (
    <div className="mt-16 mb-20">
      <h3 className="mb-2 text-xl">Pending invitations</h3>
      <span className="text-muted-foreground text-sm">
        All invitations waiting to be accepted
      </span>

            <div  className="flex  flex-col  items-center  justify-center  border  rounded-md  px-4  py-8
text-card-foreground my-8">
        {isOwnerOrManager && userOrgId ? (
          query?.invites?.isLoading ? (
            <LoadingSpinner />
          ) : query?.invites?.data && !isEmpty(query?.invites?.data) ? (
            query?.invites?.data?.map(renderItem)
          ) : (
            <div className="flex flex-col items-center justify-center gap-1 opacity-80">
              <h3 className="mb-2 text-xl">Invite Team Members</h3>
```

```
              <InviteModal currentOrganization={currentOrganization} />
            </div>
          )
        ) : (
          <div className="flex flex-col items-center justify-center gap-1 opacity-80">
            <h3 className="mb-2 text-xl">Invite Team Members</h3>
            <InviteModal currentOrganization={currentOrganization} />
          </div>
        )}
      </div>
    </div>
  );

  return null;
}

export default memo(PendingInvites);
```