```
import { UserOrganizationsProps } from './../types';
import { useToast } from '@/shared/components/ui/Toasts/use-toast';
import { useOrganizationStore } from '@/shared/stores/organization';
import { ExcludeOwner } from '../types';
import { FormEvent, SyntheticEvent, useState } from 'react';
import { emailRegExp } from '../components/team/components/const';
import { ROLES } from '@/shared/constants/organization';
import { useQueryMutation } from './organizations';

export function useInviteModal({
  currentOrganization,
}: {
  currentOrganization: UserOrganizationsProps;
}) {
  const { query, mutation } = useQueryMutation();
  const userOrgId = useOrganizationStore((state) => state.userOrgId);

  const toast = useToast();
  const [email, setEmail] = useState('');
  const [isValidEmail, setIsValidEmail] = useState(true);
  const [openDialog, setOpenDialog] = useState(false);

  const isDisabledInvite =
    !currentOrganization?.organization?.id ||
    currentOrganization?.role === 'reader';
```

```
const [inviteRole, setInviteRole] = useState<ExcludeOwner | string>(
  ROLES[ROLES.length - 1]?.value,
);

function handleEmailChange(value: string) {
  setEmail(value);
  setIsValidEmail(emailRegExp.test(value));
}

async function inviteUser(e: FormEvent) {
  e.preventDefault();

  const _email = email.trim();
  if (_email.length < 3 && !isValidEmail) {
    toast.toast({
      description: 'Enter a valid email address',
      style: { color: 'red' },
    });
    return;
  }

  if (!inviteRole) {
    toast.toast({
      description: 'Missing required values',
      style: { color: 'red' },
    });
```

```
  return;
}

useOrganizationStore.getState().setIsLoading(true);

try {
  const response = await mutation.invite.mutateAsync({
    email,
    role: inviteRole as ExcludeOwner,
    id: userOrgId ?? '',
  });
  if (response) {
    toast.toast({
      description: `${email} has been invited to join your organization.`,
      style: { color: 'green' },
    });
    setOpenDialog(false);
    query?.invites?.refetch();
  }
} catch (error) {
  console.log(error);
  if ((error as any)?.message) {
    toast.toast({
      description: (error as any)?.message,
      style: { color: 'red' },
```

```
      });
    }
  } finally {
    useOrganizationStore.getState().setIsLoading(false);
  }
}

function handleOpenModal(e: SyntheticEvent) {
  if (isDisabledInvite) {
    e.preventDefault();
    toast.toast({
      description: `Required permissions not found`,
      style: { color: 'red' },
    });
    return;
  }
}

return {
  email,
  isValidEmail,
  inviteRole,
  openDialog,
  isDisabledInvite,
  setOpenDialog,
  setInviteRole,
```

```
    inviteUser,
    handleEmailChange,
    handleOpenModal,
  };
}
```