```
import { useToast } from '@/shared/components/ui/Toasts/use-toast';
import { useOrganizationStore } from '@/shared/stores/organization';
import { debounce } from '@/shared/utils/helpers';
import { useCallback, useMemo, useState } from 'react';
import { ExcludeOwner } from '../types';
import { ROLES } from '@/shared/constants/organization';
import { useOrganizationMutation, useQueryMutation } from './organizations';
import { trpc } from '@/shared/utils/trpc/trpc';

export function useOrganizationTeam() {
  const currentOrgId = useOrganizationStore((state) => state.currentOrgId);
  const { query, mutation } = useQueryMutation();
  const { withOrganizationMutation } = useOrganizationMutation();
  const utils = trpc.useUtils();

  const toast = useToast();
  const [filterRole, setFilterRole] = useState<string>(ROLES[0].value);
  const [search, setSearch] = useState('');

  const debouncedSearch = useMemo(() => {
    const debouncedFn = debounce((value: string) => {
      setSearch(value);
    }, 100);
    return debouncedFn;
  }, []);
```

```
const isTeamMembers = query?.members?.data && query.members.data?.length >= 1;


const teamMembersToDisplay = useMemo(() => {
  if (!query?.members?.data) return [];
  return query.members.data
    .filter((member) =>
      filterRole === 'Team roles' ? true : member.role === filterRole,
    )
    .filter(
      (member) =>
        !search || member.name?.toLowerCase().includes(search.toLowerCase()),
    );
}, [query?.members?.data, filterRole, search]);


const handleSearchChange = useCallback(
  (value: string) => {
    debouncedSearch(value);
  },
  [debouncedSearch],
);


async function handleRoleChange(user_id: string, role: ExcludeOwner) {
  if (!user_id || !role) {
    toast.toast({
      description: 'Something went wrong; Missing values',
      style: { color: 'red' },
    });
    return;
  }
  await withOrganizationMutation({
    mutationFunction: mutation.changeRole,
    data: { user_id, role, organization_id: currentOrgId },
    toastMessage: 'Member role updated',
  });
}

async function handleLeaveOrg() {
  await withOrganizationMutation({
    mutationFunction: mutation.leave,
    data: { organization_id: currentOrgId },
    toastMessage: "You've successfully left the organization",
  });
  utils.organization.members.reset();
  useOrganizationStore
    .getState()
    .setCurrentOrgId(query?.organizations?.data?.[0].organization.id ?? '');
}

async function handleDeleteMember(user_id: string) {
  if (!user_id) {
    toast.toast({
      description: 'Something went wrong; Missing values',
        style: { color: 'red' },
      });
      return;
    }
    await withOrganizationMutation({
      mutationFunction: mutation.delete,
      data: { user_id, organization_id: currentOrgId },
      toastMessage: 'User has been successfully removed',
    });
  }

  return {
    search,
    filterRole,
    isTeamMembers,
    teamMembersToDisplay,
    isLoading: query?.members?.isLoading,
    setFilterRole,
    handleSearchChange,
    handleRoleChange,
    handleLeaveOrg,
    handleDeleteMember,
  };
}
```