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

import { FormEvent, useEffect, useMemo, useState } from 'react';

import { Role, UserOrganizationProps, UserOrganizationsProps } from '../types';

import { useOrganizationMutation, useQueryMutation } from './organizations';


export function useOrganizationList({

  userOrgsData,

}: {

  userOrgsData: UserOrganizationsProps[];

}) {

  const { mutation } = useQueryMutation();

  const createMutation = mutation.create;

  const updateMutation = mutation.update;


  const { handleFormMutation, openDialog, setOpenDialog } =

    useOrganizationMutation();


  const currentOrgId = useOrganizationStore((state) => state.currentOrgId);

  const userOrgId = useOrganizationStore((state) => state.userOrgId);

  const setCurrentOrgId = useOrganizationStore(

    (state) => state.setCurrentOrgId,

  );

  const [filterOrg, setFilterOrg] = useState('select-org');


  const createOrganization = (event: FormEvent<HTMLFormElement>) =>

    handleFormMutation({
```

```
    e: event,

    mutationFunction: createMutation,

    toastMessage: 'Organization has been created',

  });


const updateOrganization = (event: FormEvent<HTMLFormElement>) =>

  handleFormMutation({

    e: event,

    options: { id: userOrgId ?? '' },

    mutationFunction: updateMutation,

    toastMessage: 'Organization has been updated',

  });


const activeOrgId = useMemo(

  () =>

    userOrgsData?.find((org) => org?.organization?.id === currentOrgId)

      ?.organization?.id,

  [userOrgsData, currentOrgId],

);


// selects current organization or returns

function handleFilterOrg(value: string) {

  if (value !== 'select-org') {

    setFilterOrg(value);

    setCurrentOrgId(value);

  } else {
    setFilterOrg(activeOrgId || 'select-org');
    setCurrentOrgId(activeOrgId ?? '');
  }
}

const filteredOrg = useMemo(() => {
  if (!userOrgsData) return {};
  return userOrgsData.find((org) => org?.organization?.id === filterOrg);
}, [userOrgsData, filterOrg]) as {
  organization: UserOrganizationProps;
  role: Role;
};

// returns list of organizations to select from
// e.g [{ name: 'Select an organization', id: 'select-org' }, { name: 'Swarms', id: 'select-id' }]
const listOfOrgs = useMemo(() => {
  return userOrgsData?.reduce(
    (acc, curr) => {
      acc.push({
        name: curr?.organization?.name ?? '',
        id: curr?.organization?.id ?? '',
      });
      return acc;
    },
    [{ name: 'Select an organization', id: 'select-org' }],
  );
  }, [userOrgsData]);


  useEffect(() => {
    if (activeOrgId) {
      setFilterOrg(activeOrgId);
    } else {
      setFilterOrg('select-org');
    }
  }, [activeOrgId]);


  return {
    createOrganization,
    updateOrganization,
    handleFilterOrg,
    setOpenDialog,
    listOfOrgs,
    filteredOrg,
    filterOrg,
    openDialog,
  };
}
```