```
import { createSlice, createAsyncThunk } from "@reduxjs/toolkit";
import trainingEmployeeService from "services/Training/TrainingEmployeeService";
import { cloneDeep } from "lodash";
import { getTrainingProgramById } from "./trainingProgramSlice";

export const getTrainingEmployees = createAsyncThunk(
  "trainingEmployees/getTrainingEmployees",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await trainingEmployeeService.Get(data);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (error) {
      console.error("API call failed:", error);
      return rejectWithValue(error.message || "Error");
    }
  }
);

export const getAllTrainingEmployees = createAsyncThunk(
  "trainingEmployees/getAllTrainingEmployees",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await trainingEmployeeService.GetAll(data);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (error) {
      console.error("API call failed:", error);
      return rejectWithValue(error.message || "Error");
    }
  }
);

export const updateTrainingEmployee = createAsyncThunk(
  "trainingEmployees/updateTrainingEmployee",
  async (data, { dispatch, rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await trainingEmployeeService.Update(data);
      if (onSuccess) onSuccess(response);
      dispatch(getTrainingProgramById({ id: response.data }));
      return response.data;
    } catch (error) {
      console.error("API call failed:", error);
      return rejectWithValue(error.message || "Error");
    }
  }
);

const initialState = {
  loading: false,
  trainingEmployeeList: [],
  allEmployeeList: [],
  error: null,
};

const trainingEmployeesSlice = createSlice({
  name: "trainingEmployees",
  initialState,
  reducers: {},
  extraReducers: (builder) => {
    builder
      .addCase(getTrainingEmployees.pending, (state) => {
```

```
        state.loading = true;
        state.error = null;
      })
      .addCase(getTrainingEmployees.fulfilled, (state, action) => {
        state.loading = false;
        state.trainingEmployeeList = action.payload;
      })
      .addCase(getTrainingEmployees.rejected, (state, action) => {
        state.loading = false;
        state.error = action.payload;
      })
      .addCase(updateTrainingEmployee.pending, (state) => {
        state.loading = true;
        state.error = null;
      })
      .addCase(updateTrainingEmployee.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(updateTrainingEmployee.rejected, (state, action) => {
        state.loading = false;
        state.error = action.payload;
      })
      .addCase(getAllTrainingEmployees.pending, (state) => {
        state.loading = true;
        state.error = null;
      })
      .addCase(getAllTrainingEmployees.fulfilled, (state, action) => {
        state.loading = false;
        state.allEmployeeList = action.payload;
      })
      .addCase(getAllTrainingEmployees.rejected, (state, action) => {
        state.loading = false;
        state.error = action.payload;
      });
  },
});

export default trainingEmployeesSlice.reducer;
```