```
import { createSlice, createAsyncThunk } from "@reduxjs/toolkit";
import trainingExpenseService from "services/Training/TrainingExpenseService";
import { cloneDeep } from "lodash";

export const getTrainingExpenses = createAsyncThunk(
  "trainingExpenses/getTrainingExpenses",
  async (branchId, { rejectWithValue }) => {
    try {
      const response = await trainingExpenseService.Get(branchId);
      return response.data;
    } catch (err) {
      console.error("API call failed:", err);
      return rejectWithValue(err.message || "Error");
    }
  }
);

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

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

export const deleteTrainingExpense = createAsyncThunk(
  "trainingExpenses/deleteTrainingExpense",
  async (data, { rejectWithValue }) => {
    const { onSuccess, id } = data;
    try {
      const response = await trainingExpenseService.Delete(id);
      if (onSuccess) onSuccess(response);
      return id;
    } catch (error) {
      console.error("API call failed:", error);
      return rejectWithValue(error.message || "Error");
    }
  }
);

const initialState = {
  loading: false,
  trainingExpenseList: [],
  trainingExpenseListSetting: [],
  error: null,
```

```
};

const trainingExpensesSlice = createSlice({
  name: "trainingExpenses",
  initialState,
  reducers: {},
  extraReducers: (builder) => {
    builder
      .addCase(getTrainingExpenses.pending, (state) => {
        state.loading = true;
        state.error = null;
      })
      .addCase(getTrainingExpenses.fulfilled, (state, action) => {
        state.loading = false;
        const payload = action.payload || [];
        const filteredPayload = payload.filter(
          (item) => item.inActive === false
        );
        state.trainingExpenseList = filteredPayload;
        state.trainingExpenseListSetting = [
          {
            action: "initial",
            isRequired: true,
          },
          ...payload,
        ];
      })
      .addCase(getTrainingExpenses.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(updateTrainingExpense.pending, (state) => {
        state.loading = true;
      })
      .addCase(updateTrainingExpense.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(updateTrainingExpense.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(createTrainingExpense.pending, (state) => {
        state.loading = true;
      })
      .addCase(createTrainingExpense.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(createTrainingExpense.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(deleteTrainingExpense.pending, (state) => {
        state.loading = true;
      })
      .addCase(deleteTrainingExpense.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(deleteTrainingExpense.rejected, (state, action) => {
        state.loading = false;
      });
  },
});

export default trainingExpensesSlice.reducer;
```