```
import { createSlice, createAsyncThunk } from "@reduxjs/toolkit";
import overTimeService from "services/OverTime/OverTimeService";
import { cloneDeep } from "lodash";

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

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

export const createOverTime = createAsyncThunk(
  "overTimes/createOverTime",
  async (data, { rejectWithValue, dispatch }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await overTimeService.Create(data);
      if (onSuccess) onSuccess(response.data);
      dispatch(getOTById({ id: response.data?.id }));
      return response.data;
    } catch (error) {
      console.error("API call failed:", error);
      return rejectWithValue(error.description || "Error");
    }
  }
);

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

```
    try {
      const response = await overTimeService.GetById(id);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (error) {
      console.error("API call failed:", error);
      return rejectWithValue(error.message || "Error");
    }
  }
);

const initialState = {
  loading: false,
  overTimeList: [],
  overTimeDetail: {
    loading: false,
    data: null,
  },
  initForm: {
    employeeId: null,
    startDate: null,
    endDate: null,
    code: null,
    reason: null,
  },
};

const overTimesSlice = createSlice({
  name: "overTimes",
  initialState,
  reducers: {
    updateInitForm: (state, action) => {
      state.initForm = { ...state.initForm, ...action.payload };
    },
    resetInitForm: (state) => {
      state.initForm = initialState.initForm;
    },
    resetOTDetail: (state) => {
      state.overTimeDetail = initialState.overTimeDetail;
    },
  },
  extraReducers: (builder) => {
    builder
      .addCase(getOverTimes.pending, (state) => {
        state.loading = true;
      })
      .addCase(getOverTimes.fulfilled, (state, action) => {
        state.loading = false;
        state.overTimeList = action.payload;
      })
      .addCase(getOverTimes.rejected, (state) => {
        state.loading = false;
      })
      .addCase(updateOverTime.pending, (state) => {
        state.loading = true;
      })
      .addCase(updateOverTime.fulfilled, (state) => {
        state.loading = false;
      })
      .addCase(updateOverTime.rejected, (state) => {
        state.loading = false;
      })
      .addCase(createOverTime.pending, (state) => {
        state.loading = true;
      })
      .addCase(createOverTime.fulfilled, (state) => {
        state.loading = false;
      })
      .addCase(createOverTime.rejected, (state) => {
        state.loading = false;
      })
      .addCase(deleteOverTime.pending, (state) => {
        state.loading = true;
      })
```

```
      .addCase(deleteOverTime.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(deleteOverTime.rejected, (state) => {
        state.loading = false;
      })

      .addCase(getOTById.pending, (state) => {
        state.loading = true;
        state.overTimeDetail = { ...state.overTimeDetail, loading: true };
      })
      .addCase(getOTById.fulfilled, (state, action) => {
        state.loading = false;
        state.overTimeDetail = { data: action.payload, loading: false };
      })
      .addCase(getOTById.rejected, (state) => {
        state.loading = false;
        state.overTimeDetail = { ...state.overTimeDetail, loading: false };
      });
  },
});
export const { updateInitForm, resetInitForm, resetOTDetail } =
  overTimesSlice.actions;

export default overTimesSlice.reducer;
```