```
import { createSlice, createAsyncThunk } from "@reduxjs/toolkit";
import partnerService from "services/Partner/PartnerService";
import { cloneDeep } from "lodash";

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

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

export const updatePartner = createAsyncThunk(
  "partners/updatePartner",
  async (data, { rejectWithValue, dispatch }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await partnerService.Update(data);
      if (onSuccess) onSuccess(response);

      return response.data;
    } catch (error) {
      console.error("API call failed:", error);
      return rejectWithValue(error.message || "Error");
    }
  }
);

export const createPartner = createAsyncThunk(
  "partners/createPartner",
  async (data, { rejectWithValue, dispatch }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await partnerService.Create(data);
      if (onSuccess) onSuccess(response.data);

      return response.data;
    } catch (error) {
      console.error("API call failed:", error);
      return rejectWithValue(error.description || "Error");
    }
  }
);

export const deletePartner = createAsyncThunk(
  "partners/deletePartner",
  async (data, { rejectWithValue }) => {
    const { onSuccess, id } = data;
    try {
```

```
      const response = await partnerService.Delete(id);
      if (onSuccess) onSuccess(response);
      return id;
    } catch (error) {
      console.error("API call failed:", error);
      return rejectWithValue(error.message || "Error");
    }
  }
);
export const getPartnerById = createAsyncThunk(
  "partners/getPartnerById",
  async (data, { rejectWithValue }) => {
    const { onSuccess, id } = data;
    try {
      const response = await partnerService.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,
  partnerList: [],
  partnerDetail: {
    loading: false,
    data: null,
  },
};

const partnersSlice = createSlice({
  name: "partners",
  initialState,
  reducers: {},
  extraReducers: (builder) => {
    builder
      .addCase(getPartners.pending, (state) => {
        state.loading = true;
      })
      .addCase(getPartners.fulfilled, (state, action) => {
        state.loading = false;
        state.partnerList = action.payload;
      })
      .addCase(getPartners.rejected, (state) => {
        state.loading = false;
      })
      .addCase(updatePartner.pending, (state) => {
        state.loading = true;
      })
      .addCase(updatePartner.fulfilled, (state) => {
        state.loading = false;
      })
      .addCase(updatePartner.rejected, (state) => {
        state.loading = false;
      })
      .addCase(createPartner.pending, (state) => {
        state.loading = true;
      })
      .addCase(createPartner.fulfilled, (state) => {
        state.loading = false;
      })
      .addCase(createPartner.rejected, (state) => {
        state.loading = false;
      })
      .addCase(deletePartner.pending, (state) => {
        state.loading = true;
      })
      .addCase(deletePartner.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(deletePartner.rejected, (state) => {
```

```
        state.loading = false;
      })

      .addCase(getPartnerById.pending, (state) => {
        state.loading = true;
        state.partnerDetail = { ...state.partnerDetail, loading: true };
      })
      .addCase(getPartnerById.fulfilled, (state, action) => {
        state.loading = false;
        state.partnerDetail = { data: action.payload, loading: false };
      })
      .addCase(getPartnerById.rejected, (state) => {
        state.loading = false;
        state.partnerDetail = { ...state.partnerDetail, loading: false };
      });
  },
});
export const {} = partnersSlice.actions;

export default partnersSlice.reducer;
```