```
import { createSlice, createAsyncThunk } from "@reduxjs/toolkit";
import { cloneDeep } from "lodash";
import SettingService from "services/sale/SettingService";

export const getAllBanner = createAsyncThunk(
  "setting/getAllBanner",
  async (data, { rejectWithValue }) => {
    try {
      const response = await SettingService.searchGridBanner(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const getBannerById = createAsyncThunk(
  "setting/getBannerById",
  async (Id, { rejectWithValue }) => {
    try {
      const response = await SettingService.bannerById(Id);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const getThietLapHeThong = createAsyncThunk(
  "setting/getThietLapHeThong",
  async (Id, { rejectWithValue }) => {
    try {
      const response = await SettingService.getThietLapHeThong();
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const updateThietLapHeThong = createAsyncThunk(
  "setting/updateThietLapHeThong",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await SettingService.updateThietLapHeThong(payload);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const addNewBanner = createAsyncThunk(
  "setting/addNewBanner",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await SettingService.createBanner(payload);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
```

```
export const updateBannerApi = createAsyncThunk(
  "setting/updateBannerApi",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await SettingService.updateBanner(payload);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const deleteBannerApi = createAsyncThunk(
  "setting/deleteBannerApi",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess, id } = data;
      const response = await SettingService.deleteBanner(id);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const getAllSetting = createAsyncThunk(
  "setting/getAllSetting",
  async (data, { rejectWithValue }) => {
    try {
      const response = await SettingService.getAllSetting();
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const updateSetting = createAsyncThunk(
  "setting/updateSetting",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await SettingService.updateSetting(payload);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

const initialState = {
  loading: false,
  bannerList: [],
  bannerDetail: {},
  settingDetail: {},
  thietLapHeThongList: [],
};

export const settingSlice = createSlice({
  name: "setting",
  initialState,
  reducers: {
    showLoading: (state) => {
      state.loading = true;
    },
```

```
  },
  extraReducers: (builder) => {
    builder
      .addCase(getAllBanner.pending, (state) => {
        state.loading = true;
      })
      .addCase(getAllBanner.fulfilled, (state, action) => {
        state.loading = false;
        state.bannerList = action.payload;
      })
      .addCase(getAllBanner.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getBannerById.pending, (state) => {
        state.loading = true;
        state.bannerDetail = {};
      })
      .addCase(getBannerById.fulfilled, (state, action) => {
        state.loading = false;
        state.bannerDetail = action.payload;
      })
      .addCase(getBannerById.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(addNewBanner.pending, (state) => {
        state.loading = true;
      })
      .addCase(addNewBanner.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(addNewBanner.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(updateBannerApi.pending, (state) => {
        state.loading = true;
      })
      .addCase(updateBannerApi.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(updateBannerApi.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(deleteBannerApi.pending, (state) => {
        state.loading = true;
      })
      .addCase(deleteBannerApi.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(deleteBannerApi.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getAllSetting.pending, (state) => {
        state.loading = true;
      })
      .addCase(getAllSetting.fulfilled, (state, action) => {
        state.loading = false;
        state.settingDetail = action.payload;
      })
      .addCase(getAllSetting.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(updateSetting.pending, (state) => {
        state.loading = true;
      })
      .addCase(updateSetting.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(updateSetting.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getThietLapHeThong.pending, (state) => {
        state.loading = true;
      })
      .addCase(getThietLapHeThong.fulfilled, (state, action) => {
```

```
        state.loading = false;
        state.thietLapHeThongList = action.payload;
      })
      .addCase(getThietLapHeThong.rejected, (state, action) => {
        state.loading = false;
      });
  },
});

export const { showLoading, setRoleDetail } = settingSlice.actions;

export default settingSlice.reducer;
```