```
import { createSlice, createAsyncThunk } from "@reduxjs/toolkit";
import AttachmentService from "services/common/AttachmentService.js";

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

export const upload = createAsyncThunk(
  "attachment/upload",
  async (data, { rejectWithValue }) => {
    const { formData, onUploadSuccess, onUploadFailed } = data;
    try {
      const response = await AttachmentService.upload(formData);
      if (onUploadSuccess) onUploadSuccess(response.data);
      return response.data;
    } catch (err) {
      if (onUploadFailed) onUploadFailed();
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const createAttachment = createAsyncThunk(
  "attachment/create",
  async (data, { rejectWithValue }) => {
    try {
      const { payload, onCreateSuccess } = data;
      const response = await AttachmentService.create(payload);
      if (onCreateSuccess) onCreateSuccess(response.data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const updateAttachment = createAsyncThunk(
  "attachment/update",
  async (data, { rejectWithValue }) => {
    try {
      const { payload, onSuccess } = data;
      const response = await AttachmentService.update(payload);
      if (onSuccess) onSuccess(response.data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const deleteAttachment = createAsyncThunk(
  "attachment/delete",
  async (data, { rejectWithValue }) => {
    try {
      const { id, onSuccess } = data;
      const response = await AttachmentService.delete(id);
      if (onSuccess) onSuccess(response.data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
```

```
const initialState = {
  loading: false,
  attachmentList: [],
};

export const attachmentSlide = createSlice({
  name: "attachment",
  initialState,
  reducers: {
    showLoading: (state) => {
      state.loading = true;
    },
  },
  extraReducers: (builder) => {
    builder
      .addCase(getByObjectId.pending, (state) => {
        state.loading = true;
      })
      .addCase(getByObjectId.fulfilled, (state, action) => {
        state.loading = false;
        state.attachmentList = action.payload;
      })
      .addCase(getByObjectId.rejected, (state, action) => {
        state.loading = false;
      })

      .addCase(upload.pending, (state) => {
        state.loading = true;
      })
      .addCase(upload.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(upload.rejected, (state, action) => {
        state.loading = false;
      })

      .addCase(createAttachment.pending, (state) => {
        state.loading = true;
      })
      .addCase(createAttachment.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(createAttachment.rejected, (state, action) => {
        state.loading = false;
      })

      .addCase(updateAttachment.pending, (state) => {
        state.loading = true;
      })
      .addCase(updateAttachment.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(updateAttachment.rejected, (state, action) => {
        state.loading = false;
      })

      .addCase(deleteAttachment.pending, (state) => {
        state.loading = true;
      })
      .addCase(deleteAttachment.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(deleteAttachment.rejected, (state, action) => {
        state.loading = false;
      })
      ;

  },
});

export const { showLoading } = attachmentSlide.actions;

export default attachmentSlide.reducer;
```