```
import React, { memo, useState, useEffect } from "react";
import { Upload, message, Modal } from "antd";
import styled from "styled-components";
import { isEmpty } from "lodash";
import { LoadingOutlined, PlusOutlined } from "@ant-design/icons";
import { withFormContext } from "components/DataEntry/Form/withFormContext";
import Images from "components/Images";
import { uploadFileURL } from "store/slices/uploadFileSlice";
import { useDispatch } from "react-redux";

export default withFormContext(
  memo(function PicturesWall({
    data,
    onChange = () => null,
    disabled,
    maxImages = 8,
    onRemove,
    update_type,
    ...res
  }) {
    const dispatch = useDispatch();
    const [state, setState] = useState({
      loading: false,
      previewVisible: false,
      previewImage: "",
      previewTitle: "",
      fileList: data || [],
    });

    useEffect(() => {
      if (!isEmpty(data)) setState({ fileList: data });
    }, [data]);

    const handleCancel = () => setState({ ...state, previewVisible: false });

    const handlePreview = async (file) => {
      if (!file.url && !file.preview) {
        file.preview = await getBase64(file.originFileObj);
      }

      setState({
        ...state,
        previewImage: file.url || file.preview,
        previewVisible: true,
        previewTitle: file.name || file.url.substring(file.url.lastIndexOf("/") + 1),
      });
    };

    function getBase64(file) {
      return new Promise((resolve, reject) => {
        const reader = new FileReader();
        reader.readAsDataURL(file);
        reader.onload = () => resolve(reader.result);
        reader.onerror = (error) => reject(error);
      });
    }

    const handleChange = ({ fileList, file }) => {
      if (fileList?.length > maxImages) {
        message.error(`Tối đa ${maxImages} ảnh!`);
        return;
      }
      if (
        file.status === "removed" ||
        (file.status === "done" &&
          !fileList.some((item) => item.status === "uploading"))
      ) {
        const handleFileList = fileList.reduce((finalList, item) => {
          if (item?.response) {
            finalList.push({ ...item.response });
          } else {
            finalList.push(item);
          }
          return finalList;
        }, []);
        onChange(handleFileList);
      }
      setState({ fileList: fileList });
    };

    function beforeUpload(file) {
      const isLimit = file.size / 1024 / 1024 < 10; // Adjust size limit if needed
      if (!isLimit) {
        message.error(
          "File tải lên vượt quá dung lượng 10MB, vui lòng kiểm tra lại!"
        );
        return Upload.LIST_IGNORE;
      }
      return isLimit;
    }

    const dummyRequest = ({ file, onSuccess, onError, ...res }) => {
      const formData = new FormData();
      formData.append("fileUpload", file);
      const payload = {
        formData,
        onUploadProgress: (result) => {
          if (onSuccess) onSuccess({ url: result.data });
        },
      };
      dispatch(uploadFileURL(payload));
    };

    const { previewVisible, previewImage, fileList, previewTitle, loading } =
      state;
    const uploadButton = (
      <div>
        {loading ? (
          <LoadingOutlined />
        ) : (
          <div>
            <PlusOutlined />
            {fileList.length <= 0 && <div style={{ marginTop: 8 }}>Upload</div>}
          </div>
        )}
      </div>
    );

    return (
      <div>
        <CustomUpload
          listType="picture-card"
          className="avatar-uploader"
          fileList={fileList || []}
          customRequest={dummyRequest}
          onRemove={onRemove}
          disabled={disabled}
          beforeUpload={beforeUpload}
          button={<LoadingOutlined />}
          onChange={handleChange}
          maxCount={maxImages} // get first item if > maxImages
          onPreview={handlePreview}
          {...res}
        >
          {disabled || fileList?.length >= maxImages ? null : uploadButton}
        </CustomUpload>
        <Modal
          open={previewVisible}
          title={previewTitle}
          footer={null}
          onCancel={handleCancel}
        >
          <div className="d-flex justify-content-center">
            {previewImage && (previewImage.endsWith(".png") || previewImage.endsWith(".jpg") || previewImage.endsWith(".jpeg")) ? (
              <Images alt="example" src={previewImage} />
            ) : (
              <div>{previewTitle}</div>
            )}
          </div>
        </Modal>
      </div>
    );
  })
);

const CustomUpload = styled(Upload)`
  .ant-upload-select {
    border-color: ${({ theme }) => theme.grayBlue};
    &.ant-upload-list-item-error {
      border-color: ${({ theme }) => theme.secondary2};
  }
  .anticon {
    color: ${({ theme }) => theme.grayBlue};
    font-size: 20px;
  }
  .ant-upload-list-picture-card-container,
  .ant-upload-select-picture-card {
    width: 100px;
    height: 100px;
  }
  .ant-upload-list-item {
    border: 1px solid #e6e6e9;
    padding: 0;
    border-radius: 4px;
  }
  img {
    border-radius: 3px;
  }
`;
```