```
import React, { memo, useState, useEffect } from "react";
import { Upload, message, Image } 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 { useDispatch } from "react-redux";
import { uploadFileURL } from "store/slices/uploadFileSlice";
export default withFormContext(
 memo(function Avatar({
    data.
    onChange,
    disabled,
   name = "avatarUrl",
    folder = null,
 }) {
    const dispatch = useDispatch();
    const [state, setState] = useState({
      loading: false,
      imageUrl: data?.location?.replace("http:", "https:") || "",
    });
    useEffect(() => {
      if (!isEmpty(data))
        setState({ imageUrl: data?.location?.replace("http:", "https:") });
    }, [data]);
    function getBase64(img, callback) {
      const reader = new FileReader();
      reader.addEventListener("load", () => callback(reader.result));
      reader.readAsDataURL(img);
    const handleChange = (info) => {
      // let fileList = [...info.fileList];
      if (info.file.status === "uploading") {
        setState({ loading: true });
        return;
      if (info.file.status === "error") {
        // Get this url from response in real world.
        getBase64(info.file.originFileObj, (imageUrl) =>
          setState({
            imageUrl,
            loading: false,
          })
        );
      if (info.file.status === "done") {
        // Get this url from response in real world.
        setState({
          imageUrl: info.file.response,
          loading: false,
        });
        if (onChange) onChange(info.file.response);
      }
    };
    function beforeUpload(file) {
      const isJpgOrPng =
        (file.type === "image/jpeg" || file.type === "image/png") &&
        /.png|.jpeg|.jpg/.test(file.name);
      if (!isJpgOrPng) {
  message.error("You can only upload JPG/PNG file!");
      }
      const isLimit = file.size / 1024 / 1024 < 4;</pre>
      if (!isLimit) {
        message.error(
          "File tải lên vượt quá dung lượng 2MB, vui lòng kiểm tra lại!"
        );
```

```
return isJpgOrPng && isLimit;
    const { imageUrl, loading } = state;
    const uploadButton = (
      <div>{loading ? <LoadingOutlined /> : <PlusOutlined />}</div>
    const dummyRequest = ({ file, onSuccess }) => {
      const formData = new FormData();
      formData.append("fileUpload", file);
      !isEmpty(folder) && formData.append("nameFolder", folder);
      const payload = {
        formData,
       onUploadProgress: (result) => {
          if (onSuccess) onSuccess(result.data);
       },
      dispatch(uploadFileURL(payload));
    };
    return (
      <div>
        <CustomUpload
          name={name}
          listType="picture-card"
          iconRender
          hasImage={!!imageUrl}
          className="avatar-uploader"
          showUploadList={false}
          customRequest={dummyRequest}
          disabled={disabled}
          beforeUpload={beforeUpload}
          onChange={handleChange}
          {imageUrl ? <Image src={imageUrl} alt="avatar" /> : uploadButton}
        </CustomUpload>
      </div>
    );
 })
const CustomUpload = styled(Upload)`
  .ant-upload-select {
   width: 100%;
   margin: 0 auto;
   height: auto;
    border-width: ${({ hasImage }) => (hasImage ? "0px" : "1px")};
    border-radius: 4px;
    border-color: ${({ theme }) => theme.grayBlue};
    span.ant-upload {
      color: ${({ theme }) => theme.grayBlue};
      font-size: 20px;
      position: relative;
      display: block;
      padding-bottom: 100%;
      width: 100%;
      height: 0px;
      > * {
       top: 50%;
        left: 50%;
       width: 100%;
        position: absolute;
        transform: translate(-50%, -50%);
      }
      img {
        border-radius: 4px;
       max-width: 100%;
       max-height: 230px;
        object-fit: contain;
     }
   }
```