Image overlaping on React

<u>Video send by admin using multer :-</u>

```
import axios from 'axios';
import React, { useEffect, useState } from 'react';
import { Button, Input, Label } from 'reactstrap';
import { useNavigate } from 'react-router-dom';

const Forward = () => {

  //Store the video...
  const [data, setData] = useState(null);

  //After video send redirect to overlap page..
  const navigate = useNavigate();

  //Handle the form data...
  const handlesubmit = async (e) => {
    const formdata = new FormData();
    formdata.append('file', data);
    const response = await axios.post("/api/auth/upload", formdata);
    if (response) {
      console.log("Video send using multer...");
      navigate("/temp");
    } else {
      console.log("Video is not sent!");
    }
  }

  return (
    <div>
      <Label for="exampleFile">
        Selected Video :-
      </Label>
      <br />
      <Input
        id="exampleFile"
        name="file"
        type="file"
        onChange={(e) => setData(e.target.files[0])}
      /><br />
      <Button onClick={handlesubmit} color='info'>Submit</Button>
    </div>
  )
}

export default Forward;
```

Controller to work store and fetch(display)n the video..

```
const videoModel = require('../models/videoModel');

//Store the video into database...
const videoController = async (req, res) => {
    try {
        await videoModel.create({
            video: req.file.filename
        });
        res.status(201).send({ message: "Data send to server." });
    } catch (err) {
        console.log(err);
        res.status(501).send({ message: "Data is not send." })
    }
}

//Fetch the video from database...
const videoFetch = async (req, res) => {
    try {
        const data = await videoModel.find({});

        if (data) {
            //return res.status(201).json({ message: "Videos fetch
successfully.", data: data });
            const vidoeName = data[0]?.video
            return res.status(201).json(vidoeName)
        } else {
            return res.status(501).json({ message: "Videos is not found." });
        }
    } catch (err) {
        console.log(err);
    }
}

module.exports = {
    videoController,
    videoFetch
}
```

<u>Now get the video using multer, and overlap the image on video...</u>

```
import React, { useRef, useState, useEffect } from 'react';
//import videoSource from "../data/tamp1.mp4";
import axios from 'axios';

const VideoWatermark = () => {
    const videoRef = useRef(null);
    const [showWatermark, setShowWatermark] = useState(false);
    const [watermarkImage, setWatermarkImage] = useState(null);
    const [fetchVideo, setFetchVideo] = useState("");

    const handleImageChange = (e) => {
        const file = e.target.files[0];
        if (file) {
            const reader = new FileReader();
            reader.onloadend = () => {
                setWatermarkImage(reader.result);
            };
            reader.readAsDataURL(file);
        }
    };

    const addWatermark = () => {
        setShowWatermark(true);
    };

    const downloadWithWatermark = async () => {
        const video = videoRef.current;
        const canvas = document.createElement('canvas');
        canvas.width = video.videoWidth;
        canvas.height = video.videoHeight;
        const ctx = canvas.getContext('2d');
        const stream = canvas.captureStream();
        const recorder = new MediaRecorder(stream, { mimeType: 
'video/webm;codecs=vp9' });
        const chunks = [];

        recorder.ondataavailable = (event) => {
            if (event.data.size > 0) {
                chunks.push(event.data);
            }
        };

        recorder.onstop = () => {
            const blob = new Blob(chunks, { type: 'video/webm' });

            const downloadLink = document.createElement('a');
            downloadLink.href = URL.createObjectURL(blob);
            downloadLink.download = 'video_with_watermark.mkv';  // Change the 
file extension to .mkv
            downloadLink.click();
        };

        recorder.start();

        video.play();

        const drawFrame = () => {
            ctx.clearRect(0, 0, canvas.width, canvas.height);
            ctx.drawImage(video, 0, 0, canvas.width, canvas.height);

            if (showWatermark && watermarkImage) {
                const watermark = new Image();
                watermark.src = watermarkImage;

                const watermarkWidth = 150; // Set the desired width for the 
circular watermark
                const watermarkHeight = watermarkWidth;

                const watermarkPosition = { x: 10, y: canvas.height - 10 - 
watermarkHeight };

                // Save the current context state
                ctx.save();

                // Create a circular mask
                ctx.beginPath();
                ctx.arc(watermarkPosition.x + watermarkWidth / 2,
watermarkPosition.y + watermarkHeight / 2, watermarkWidth / 2, 0, 2 * Math.PI);
                ctx.closePath();
                ctx.clip();

                // Draw the circular watermark
                ctx.drawImage(watermark, watermarkPosition.x,
watermarkPosition.y, watermarkWidth, watermarkHeight);

                // Restore the context to its previous state
                ctx.restore();
            }

            if (!video.ended) {
                requestAnimationFrame(drawFrame);
            }
        };

        drawFrame();

        setTimeout(() => {
            recorder.stop();
            video.pause();
        }, video.duration * 2000);
    };

    //Fetch the video from server...
    useEffect(() => {
        axios.get("/api/auth/pullVideo")
            .then(res => {
                console.log(res.data);
                setFetchVideo(res.data);
            })
            .catch(err => console.log(err))
    }, []);

    return (
        <div style={{ position: 'relative' }}>
            {fetchVideo && (
                <video ref={videoRef} controls width="500">
                    <source src={`http://localhost:8050/${fetchVideo}`}
type="video/mp4" />
                    Your browser does not support the video tag.
                </video>
            )}
            {showWatermark && watermarkImage && (
                <div style={{ position: 'absolute', bottom: '10px', left: '10px',
borderRadius: '50%', overflow: 'hidden' }}>
                    <img src={watermarkImage} alt="Watermark" style={{ maxWidth:
'100px', borderRadius: '50%' }} />
                </div>
            )}
            <input type="file" accept="image/*" onChange={handleImageChange} />
            <button onClick={addWatermark}>Add Watermark</button>
            <button onClick={downloadWithWatermark}>Download with
Watermark</button><br />
            {/* <video src={`http://localhost:8050/${fetchVideo}`} controls
width="500" /> */}
        </div>
    );

};

export default VideoWatermark;
```