SNAPCHAT:

Dependencies:

yarn add [firebase@^8.10.0](mailto:firebase@%5e8.10.0)

yarn add react-webcam (camera)

yarn add react-router-dom

yarn add uuid

yarn add react-timeago

yarn add react-countdown-circle-timer (for countdown)

Webcam feature:

import './WebCamCapture.css'

import Webcam from 'react-webcam'

import { useCallback, useRef, useState } from 'react'

import {RadioButtonUnchecked} from '@material-ui/icons'

const videoConstraints = {

    width: 250,

    height: 400,

    facingMode: "user"

}

function WebCamCapture() {

    const webcamRef = useRef()

    const [image, setImage] = useState(null)

    const capture = useCallback(()=>{

        const imageSrc = webcamRef.current.getScreenshot()

        setImage(imageSrc)

    }, [webcamRef])

    return (

        <div className="webcamCapture">

            <Webcam

                audio={false}

                height={videoConstraints.height}

                ref={webcamRef}

                screenshotFormat='image/jpeg'

                width={videoConstraints.width}

                videoConstraints={videoConstraints}

            />

            <RadioButtonUnchecked

                className="webcamcapture\_\_icons"

                onClick={capture}

                fontSize="large"

            />

            <img src={image}/>

        </div>

    )

}

export default WebCamCapture

Advanced if needed:

import './WebCamCapture.css'

import Webcam from 'react-webcam'

import { useCallback, useRef, useState } from 'react'

import {RadioButtonUnchecked} from '@material-ui/icons'

import { useDispatch } from 'react-redux'

import { setCameraImage } from '../../features/cameraSlice'

import { useHistory } from 'react-router'

const videoConstraints = {

    width: 250,

    height: 400,

    facingMode: "user"

}

function WebCamCapture() {

    const webcamRef = useRef()

    const dispatch = useDispatch()

    const history = useHistory()

    const capture = useCallback(()=>{

        const imageSrc = webcamRef.current.getScreenshot()

        dispatch(setCameraImage(imageSrc));

        history.push('/preview')

    }, [webcamRef])

    return (

        <div className="webcamCapture">

            <Webcam

                audio={false}

                height={videoConstraints.height}

                ref={webcamRef}

                screenshotFormat='image/jpeg'

                width={videoConstraints.width}

                videoConstraints={videoConstraints}

            />

            <RadioButtonUnchecked

                className="webcamcapture\_\_icons"

                onClick={capture}

                fontSize="large"

            />

        </div>

    )

}

export default WebCamCapture

Importing WebcamCapture.js

import React from 'react';

import './App.css';

import WebCamCapture from './components/WebCamCapture/WebCamCapture';

import {

  BrowserRouter as Router,

  Switch,

  Route,

  Link

} from "react-router-dom";

import Preview from './components/Preview/Preview';

function App() {

  return (

    <div className="app">

      <Router>

      <div className="app\_\_body">

        <Switch>

          <Route exact path="/">

            <WebCamCapture/>

          </Route>

          <Route path="/preview">

            <Preview/>

          </Route>

        </Switch>

      </div>

    </Router>

    </div>

  );

}

export default App;

And it is working on browser now

Uploading image to firebase storage and creating collection of firestore containing image link and some extra stuff

Getting data from firestore

Updating Data in firebase firestore using set:

Merge:true is used to merge update with all data that was their, it will not over write any other data

import { Avatar } from '@material-ui/core'

import { StopRounded } from '@material-ui/icons'

import { useDispatch, useSelector } from 'react-redux'

import ReactTimeago from 'react-timeago'

import { selectImage } from '../../../features/appSlice'

import './Chat.css'

import db from '../../../firebase'

import { useHistory } from 'react-router'

function Chat({ id, username, timestamp, read, imageUrl, profilePic }) {

    const dispatch = useDispatch();

    const history = useHistory()

    const open = () => {

        if (!read){

            dispatch(selectImage(imageUrl))

            db.collection('posts').doc(id).set({

                read: true,

            }, { merge: true });

            history.push('/chats/view');

        }

    }

    return (

        <div onClick={open} className="chat">

            <Avatar className="chat\_\_avatar" src={profilePic}/>

            <div className="chat\_\_info">

                <h4>{username}</h4>

                <p>Tap to view - <ReactTimeago date={new Date(timestamp?.toDate()).toUTCString()}/></p>

            </div>

            {!read && <StopRounded className="chat\_\_readIcon"/>}

        </div>

    )

}

export default Chat

CountDown Timer:

import { useEffect } from 'react'

import { useSelector } from 'react-redux'

import { useHistory } from 'react-router'

import './ChatView.css'

import {CountdownCircleTimer} from 'react-countdown-circle-timer'

function ChatView() {

    const selectedImage = useSelector(state=>state.app.selectedImage)

    const history = useHistory()

    useEffect(()=>{

        if (!selectedImage){

            exit();

        }

    },[selectedImage])

    const exit = () => {

        history.replace('/chats')

    }

    return (

        <div className="chatView">

            <img src={selectedImage} onClick={exit} alt="selected\_image"/>

            <div className="chatView\_\_timer">

                <CountdownCircleTimer

                    isPlaying

                    duration={10}

                    strokeWidth={6}

                    size={50}

                    colors={[

                        ["#004777", 0.33],

                        ["#F7B801", 0.33],

                        ["#A30000", 0.33],

                    ]}

                >

                    {({remainingTime})=>{

                        if(remainingTime===0){

                            exit();

                        }

                        return remainingTime

                    }}

                </CountdownCircleTimer>

            </div>

        </div>

    )

}

export default ChatView

Its css:

.chatView{

    position: relative;

}

.chatView > img {

    cursor: pointer;

}

.chatView\_\_timer{

    position: absolute;

    top: 0;

    right: 0;

    margin: 10px;

}

Absolute with translate:

.webcamCapture{

    position: relative

}

.webcamcapture\_\_button{

    position: absolute;

    bottom: 0;

    left: 50%;

    transform: translate(-50%, -50%);

    cursor: pointer;

    color: red;

}

Hiding Scrollbar but having scrollbar functionality working:

.chats\_\_posts{

    height: 359px;

    background-color: white;

    box-shadow: 1px -7px 7px -6px rgba(0,0,0,0.44);

    margin-top: -9px;

    border-top-left-radius: 10px;

    border-top-right-radius: 10px;

    background-color: white;

    overflow: scroll;

}

/\*rid from overflow scrollbar\*/

.chats\_\_posts::-webkit-scrollbar{

    display: none;

}

/\*hiding scroll bar for edge, IE, firefox \*/

.chats\_\_posts{

    -ms-overflow-style: none;

    scrollbar-width: none;

}