# Internet Technology and Applications Practicals
Assignment 4

Krunal Rank
U18CO081

Create a simple CRUD app that interacts with a database. UI must be very clean and simple and it must be responsive.

**Tech Stack**

ReactJS
FirebaseSDK
AlgoliaSearch
MaterialUI

**Project Directory Structure**

```
./
├── App.css
├── App.js
├── App.scss
├── assets
│   ├── google_logo.png
│   └── NIT_Surat.png
├── components
│   ├── CreatePost
│   │   └── index.js
│   ├── Footer
│   │   └── index.js
│   ├── GuestLinks
│   │   └── index.js
│   ├── Navbar
│   │   └── index.js
│   ├── policy
│   │   └── index.js
│   ├── Post
│   │   └── index.js
│   ├── SignInLinks
│   │   └── index.js
│   └── UserView
│       └── index.js
├── config
│   └── firebaseConfig.js
├── index.css
├── index.js
├── logo.png
├── logo.svg
├── registerServiceWorker.js
├── reportWebVitals.js
├── routes
│   ├── Dashboard
│   │   └── index.js
│   ├── Home
│   │   └── index.js
│   ├── OurStory
│   │   └── index.js
│   ├── Profile
│   │   └── index.js
│   └── SearchView
│       └── index.js
├── setupTests.js
└── utils
    └── compress.js
```

**Code:**

**App.scss:**

```
$xs: 0px;
$sm: 600px;
$md: 960px;
$lg: 1280px;
$xl: 1920px;

$spacing: 8;

a{
 text-decoration: none;
 color: inherit;
 &:hover{
   text-decoration: none;
 }
}

.search {
 position:relative;
 border-radius: 4px;
 background: rgba(255,255,255,0.15);
 &:hover {
   background: rgba(255,255,255,0.25);
 };
 margin-right: 16px;
 width: 100%;
 @media screen and (min-width: $sm) {
```

```scss
   margin-right: 24px;
   width: auto;
 }
}

.searchIcon {
 padding: 6px;
 height: 100%;
 position: absolute;
 pointer-events: none;
 display: flex;
 align-items: center;
 justify-content: center;
}

.inputBase {
 color: inherit;
 padding: 4px;
 padding-left: 33px;
 width: 100%;
 @media screen and (min-width: $sm){
  input{
   transition: 0.5s all ease-in-out;
   width: 20ch;
   &:focus {
     width: 30ch;
   }
  }
 }
}

.grow {
 flex-grow: 1;
}

.navbar {
 background-color: black;
}



.menuButton {
 margin-right: '16px';
}
```

```
.iconDesktop {
 width: 0rem;
 display: flex;
 opacity: 0;
 transition: all 0.5s ease-in-out;

 @media screen and (min-width: $sm) {
   display: flex;
   opacity: 1;
   width: 2.5rem;
   margin-right: 0.5rem;
 }
}



.react-html5-camera-photo {
 video {
   width: 100%;
 }
}

.title {
 flex-grow: 1;
 color: white;
 text-decoration: none;
 display: block;
 transition: 1s all ease-in-out;

 &:hover {
   color: 'white';
   text-decoration: 'none';
 }


}



.sectionDesktop {
 &>* {
   margin: #{$spacing*1px};
 }

  display: none;
 @media screen and (min-width: $sm) {
   display: flex;
 }
}



.navButtons *+* {
 margin: 0.25rem;
}

.list {
 width: '250px';
}

.drawerPhoto {
 width: 4rem;
 height: 4rem;
}

.block {
 display: block;
}

.progressGrid {
 display: flex;
 justify-content: center;
}


/* Home SCSS */

.home {
 margin-top: 5rem;
}

.homeTitle {
 font-size: 4rem;
}

.homeTitlePlaceholder {
 width: 15rem;
 height: 3rem;
}
```

```
.homeCenter {
 display: flex;
 justify-content: center;
 margin: 1rem;
}

.homeLogo {
 width: 300px;
}

.homeLogoPlaceholder {
 width: 300px;
 height: 300px;
}

.signInButton {
 background-color: #4285F4;
}

.homeButtonPlaceholder {
 width: 15rem;
 height: 2rem;
}

.homeGoogleLogo {
 width: 2rem;
 margin-right: 1rem;
}

/* Footer SASS */

.footer {
 margin-top: 7rem;
}

.footerCenter {
 display: flex;
 justify-content: center;
 margin: 1rem;
 text-align: center;

 * {
   text-align: center;
 }

 a {
   color: whitesmoke;
   text-decoration: none;
   margin: 1rem;
   font-size: 0.65rem;
 }
}

/* Dashboard SASS */

.dashboard {
 margin-top: 5rem;

 @media screen and (min-width: $md) {
   margin-left: 20rem;
   margin-right: 20rem;
 }
}

.dashboardCreatePostPlaceholder {
 height: 6rem;
}


/* CreatePost SASS */

.createPost {
 padding: 1rem;

}



.createPostTextField {
 width: 100%;
}

.createPostFAB {
 width: 2.5rem;
 height: 2.5rem;
 margin: 1rem;
}

.createPostHeader {
 align-items: center;
}

.createPostPreview {
 p {
   word-break: break-all;
 }
}

.createPostCardImage {
 width: 100%;
 height: 100%;
}

.cameraDialogContent {
 display: contents;
}

.cameraDialogTitle {
 padding-bottom: 0.25rem;
 padding-top: 0.25rem;
}


/* Post SASS */

.postPlaceholder {
 height: 15rem;
}

.postPaper {
 margin-top: 2rem;
 margin-bottom: 2rem;
}

/* Profile SASS */

.profile {
 margin-top: 5rem;

 @media screen and (min-width: $md) {
   margin-left: 14rem;
   margin-right: 14rem;
 }
}

.profileTop {
  & > div {
   text-align: center;
   padding-top: 0.3rem;
   padding-bottom: 0.3rem;
 }
 }

.profileTabView {
  & > div {
   text-align: center;
   padding-top: 0.3rem;
   padding-bottom: 0.3rem;
 }
 }

.profilePlaceholder {
 height: 16rem
}

.profilePhoto {
 width: 100px;
 height: 100px;
}

/* UserView SASS */

.userViewPlaceholder{
 height: 14rem;
}

.userViewLink {
 text-decoration: none;
 &:hover{
   text-decoration: none;
 }
}
```

**CreatePost/index.js:**

```
import { Component } from "react"
import { withRouter } from 'react-router-dom'
import { IconButton, CardHeader, Card, CardActions, CardContent, Container, Grid, Paper,
TextField, Box, Fab, Tooltip, Zoom, Divider, Avatar, Typography, CircularProgress,
Dialog, DialogContent, DialogTitle } from '@material-ui/core'
import firebase from 'firebase'
import { Send, Publish, MoreVert, Favorite, Cancel } from '@material-ui/icons'
import CameraIcon from '@material-ui/icons/Camera'
import Camera from 'react-html5-camera-photo'
import reduceFileSize from '../../utils/compress'



class CreatePost extends Component {

 constructor(props) {
   super(props)
   this.state = {
     user: this.props.user,
     imageUrl: '',
     postContent: '',
     sendingPost: false,
     isProfilePicModalOpen: false,
     isPhotoModalOpen: false,
     isCameraModalOpen: false,
     videoStream: null
   }
 }

 preventDefault = (e) => {
   e.preventDefault()
 }

 handleTextFieldChange = (e) => {
   if (e.target.value.length > 200) return;
   this.setState({
     ...this.state,
     postContent: e.target.value,
   })
 }

 onFileUpload = (e) => {
   let file = e.target.files[0]
   reduceFileSize(file, 500 * 1024, 1000, Infinity, 0.6, (file) => {
     let reader = new FileReader()
     reader.readAsDataURL(file)
     let flag = 0
     reader.onload = () => {
       if (!reader.result.startsWith('data:image'))
         flag = 1
       else {
         this.setState({
           ...this.state,
           imageUrl: reader.result
         })
       }
     };
     if (flag) {
       return
     }
   })

 }

 toggleProfilePicModal = () => {
   this.setState({
     ...this.state,
     isProfilePicModalOpen: !this.state.isProfilePicModalOpen
   })
 }

 togglePhotoModal = () => {
   this.setState({
     ...this.state,
     isPhotoModalOpen: !this.state.isPhotoModalOpen
   })
 }

 toggleCameraModal = async () => {
   await this.setState({
     ...this.state,
     isCameraModalOpen: !this.state.isCameraModalOpen
   })
   if (!this.state.isCameraModalOpen) {
     await Promise.all(this.state.videoStream.getTracks().map(async (t) => {
       console.log(t)
       await t.stop();
     }))
     this.setState({
       ...this.state,
       videoStream: null
     })
   }
 }

 handleUpload = async () => {
   const fileSelector = document.getElementById('fileUpload')
   fileSelector.click()
 }


 handleCameraStart = (stream) => {
   this.setState({
     ...this.state,
     videoStream: stream
   })
 }

 takeSnapPhoto = async () => {
   const videoElement = document.getElementsByTagName('video')[0]
   const canvas = document.createElement("canvas");
   canvas.width = videoElement.videoWidth;
   canvas.height = videoElement.videoHeight;
   canvas.getContext('2d')
     .drawImage(videoElement, 0, 0, canvas.width, canvas.height);
   const dataURL = canvas.toDataURL();
   await this.setState({
     imageUrl: dataURL,
   })
   this.toggleCameraModal()
 }


 getTime = (timestamp) => {
   if (this.state.isLoading) return ''
   let d = timestamp
   let currentTime = new Date().getTime()

   const monthNames = ["January", "February", "March", "April", "May", "June",
     "July", "August", "September", "October", "November", "December"
   ];
   const min = 60 * 1000
   const hour = 60 * min
   if (currentTime - d <= 15 * min) return 'Few mins ago'
   else if (currentTime - d <= 30 * min) return 'Half hour ago'
   else if (currentTime - d <= hour) return 'An hour ago'
```

```
   else if (currentTime - d <= 2 * hour) return '2 hours ago'
   else if (currentTime - d <= 4 * hour) return '4 hours ago'
   else if (currentTime - d <= 8 * hour) return '8 hours ago'
   else if (currentTime - d <= 12 * hour) return '12 hours ago'
   else if (currentTime - d <= 24 * hour) return 'A day ago'

   d = new Date(d)
   return monthNames[d.getMonth()] + ' ' + d.getDate() + ', ' + d.getFullYear()

 }

 createPostMethod = async () => {
   if (this.state.imageUrl.length === 0) {
     return
   }
   this.setState({
     ...this.state,
     sendingPost: true
   })

   try {
     let tags = this.state.postContent.split(/[\n\t ]+/).filter((e) => { return
e.startsWith('#') }).map((e) => { return e.split('#') })
     tags = [].concat.apply([], tags).filter((e) => { return e.length !== 0 }).map((e) =>
{ return e.toLowerCase() }).filter((v, i, a) => a.indexOf(v) === i)

     let dbRef = firebase.firestore()

     let postData = {
       byID: this.state.user.uid,
       imageUrl: this.state.imageUrl,
       likes: [],
       postContent: this.state.postContent,
       tags: tags,
       downloads: 0,
       timestamp: new Date().getTime(),
     }

     let postResult = await dbRef.collection('posts').add(postData)

     let postID = postResult.id.toString()

     await Promise.all(tags.map(async (tag) => {
       let tagRef = dbRef.collection('tags').doc(tag)
       let tagResult = await tagRef.get()
       if (!tagResult.exists) {
         await tagRef.set({
           posts: [postID],
           search: 0,
         })
       } else {
         let data = tagResult.data()
         tagRef.set({
           posts: [...data.posts, postID]
         })
       }
     }))

     this.setState({
       postContent: '',
       imageUrl: '',
       imageFile: null,
       sendingPost: false
     })
   } catch (e) {
     console.log(e)
   }



 }

 render() {
   if (this.state.sendingPost)
     return (
       <Grid item className='progressGrid' xs={12}>
         <CircularProgress color='secondary' />
       </Grid>
     )
   return (
     <Paper elevation={0}>
       <input hidden type="file" id='fileUpload' accept='.jpg,.jpeg,.png'
onChange={this.onFileUpload} />
       <Grid className='createPost'>
         <Grid item xs={12}>
           <TextField
             className='createPostTextField'
             label="Write your Story"
             id="outlined-size-normal"
             variant="outlined"
             color='secondary'
             multiline
```

```
             onChange={this.handleTextFieldChange}
             value={this.state.postContent}

             rowsMax='3'
           />
           <Box display='flex' flexDirection='row-reverse'>
             <p>{200 - this.state.postContent.length} characters left.</p>
           </Box>
         </Grid>
         <Grid item xs={12}>

           <Box display='flex' flexDirection="row-reverse">
             <Tooltip TransitionComponent={Zoom} title="Create Post" aria-label="Create
Post" arrow>
               <Fab color="secondary" className='createPostFAB' size='small'
onClick={this.createPostMethod} >
                 <Send />
               </Fab>
             </Tooltip>
             <Tooltip TransitionComponent={Zoom} title="Upload Photo" aria-label="Upload
Photo" arrow>

               <Fab color="secondary" className='createPostFAB' size='small'
onClick={this.handleUpload} >
                 <Publish />
               </Fab>
             </Tooltip>
             <Tooltip TransitionComponent={Zoom} title="Snap Photo" aria-label="Snap
Photo" arrow>
               <Fab color="secondary" className='createPostFAB' size='small'
onClick={this.toggleCameraModal} >
                 <CameraIcon />

               </Fab>
             </Tooltip>
           </Box>
         </Grid>
         <Divider />
         {
           (this.state.postContent.length !== 0 || this.state.imageUrl.length !== 0)
             ?
             <Grid item xs={12} className='createPostPreview'>
               <Grid item xs={12}>
                 <Typography>Post Preview</Typography>
               </Grid>
               <Card variant='outlined' className='createPostCard'>
```

```
                    <CardHeader
                      avatar={
                        <Avatar alt={this.state.user.displayName}
src={this.state.user.photoURL.replace('s96-c', 's500-c')}
onClick={this.toggleProfilePicModal} />
                      }
                      action={
                        <IconButton aria-label="settings">
                          <MoreVert />
                        </IconButton>
                      }
                      title={this.state.user.displayName}
                      subheader={this.getTime(new Date().getTime())}
                    />
                    <CardContent>
                      {this.state.postContent.split('\n').map((e) => { return <p>{e}</p> })}
                    </CardContent>
                    {
                      this.state.imageUrl.length !== 0
                        ?
                        <div>
                          <img alt='Upload'
                            className='createPostCardImage'
                            src={this.state.imageUrl} onClick={this.togglePhotoModal}
                          />
                        </div>

                        :
                        null

                    }
                    <CardActions disableSpacing>
                      <IconButton aria-label="Like Post">
                        <Favorite />
                      </IconButton>
                    </CardActions>
                  </Card>
                </Grid>
                :
                null
            }
          </Grid>
          <Dialog className='dialog'
            aria-labelledby='Profile Photo Dialog'
            aria-describedby='Profile Photo' onClose={this.toggleProfilePicModal}
open={this.state.isProfilePicModalOpen}>
```

```
         <img src={this.state.user.photoURL.replace('s96-c', 's500-c')} alt='Profile' />
       </Dialog>
       <Dialog className='dialog'
         aria-labelledby='Photo Dialog'
         aria-describedby='Photo' onClose={this.togglePhotoModal}
open={this.state.isPhotoModalOpen}>
         <img src={this.state.imageUrl} alt='Upload' />
       </Dialog>
       <Dialog className='dialog'
         aria-labelledby='Camera Photo Dialog'
         aria-describedby='Camera Photo' onClose={this.toggleCameraModal}
onBackdropClick="false" onClick={this.preventDefault}
open={this.state.isCameraModalOpen}>
         <DialogTitle className='cameraDialogTitle'>

           <Box display='flex' flexDirection='row-reverse' justifyContent='space-between'
alignItems='center'>
             <IconButton onClick={this.toggleCameraModal}><Cancel /></IconButton>
             <div>
               Snap your Photo</div>
           </Box>
         </DialogTitle>
         <DialogContent className='cameraDialogContent'>

           <Camera isImageMirror={false}
             onCameraStart={this.handleCameraStart} />
           <Tooltip TransitionComponent={Zoom} title="Snap Photo" aria-label="Snap Photo"
arrow>
             <Fab color="secondary" className='createPostFAB' size='small'
onClick={this.takeSnapPhoto} variant='extended' >
               <CameraIcon />
                   Snap
                 </Fab>
           </Tooltip>
         </DialogContent>
       </Dialog>
     </Paper>

   )
 }

}

export default withRouter(CreatePost);
```

**Footer/index.js:**

```
import { Component } from "react"
import { Link, withRouter } from 'react-router-dom'
import { Grid, Dialog, DialogContent,DialogTitle,Typography } from '@material-ui/core'
import policy from '../policy'
var template = { __html: policy };

class Footer extends Component {

 constructor(props) {
   super(props)
   this.state = {
     openPolicy: false,
     openHelp: false,
   }
 }
 preventDefault = (event) => event.preventDefault();
 handlePolicyModal = (event, reason) => {
   if (reason === 'clickaway') {
     return
   }
   this.setState({
     ...this.state,
     openPolicy: !this.state.openPolicy,
   })
 }
 handleHelpModal = (event, reason) => {
   if (reason === 'clickaway') {
     return
   }
   this.setState({
     ...this.state,
     openHelp: !this.state.openHelp,
   })
 }

 render() {
   return (
     <Grid container className='footer'>
       <Grid container className='footerCenter'>
         <Grid item xs={3} sm={1}>
           <Link to='/ourstory'>About</Link>
         </Grid>
         <Grid item xs={3} sm={1}>
           <Link to='/'>Blog</Link>
         </Grid>
```

```
          <Grid item xs={3} sm={1}>
            <Link onClick={this.handleHelpModal}>Help</Link>
          </Grid>
          <Grid item xs={3} sm={1}>
            <Link to='/'>Jobs</Link>
          </Grid>
          <Grid item xs={3} sm={1}>
            <Link to='/'>API</Link>
          </Grid>
          <Grid item xs={3} sm={1}>
            <Link to='/hashtags'>Hashtags</Link>
          </Grid>
          <Grid item xs={3} sm={1}>
            <Link to='/'>Locations</Link>
          </Grid>
          <Grid item xs={3} sm={1}>
            <Link to='/accounts'>Accounts</Link>
          </Grid>
          <Grid item xs={12}>
            <Link onClick={this.handlePolicyModal}>©Copyright 2021 - Present</Link>
          </Grid>
          <Grid item xs={12}>
            <Link onClick={this.handlePolicyModal}>All Rights Reserved.</Link>
          </Grid>
          <Grid item xs={12}>
            <Link to='/'>Fotorama</Link>
          </Grid>
        </Grid>

        <Dialog className='dialog'
              aria-labelledby="alert-dialog-title"
              aria-describedby="alert-dialog-description" onClick={this.handleHelpModal}
open={this.state.openHelp}>

              <DialogTitle id="alert-dialog-title">Fotorama | Help & FAQ</DialogTitle>
              <DialogContent>
                <Typography variant="h6">What is Fotorama all about?</Typography>
                <Typography >
                  Fotorama is what you all think it is! Snap your Pictures, Share them
with your Loved Ones, friends and families! Cherish your memories!
                </Typography>
                <Typography variant="h6">What Information do you store about
me?</Typography>
                <Typography >
                  Apart from the content that you post on Fotorama, nothing else.
                </Typography>
                <Typography variant="h6">What more can I do after logging in?</Typography>
                <Typography >
                  You can share your snaps, make friends, search posts based on Tags and
more!
                </Typography>
                <Typography variant="h6">How do I post?</Typography>
                <Typography >
                  Do you see that Share button on your Dashboard? Click it and find the
rest of the steps!
                </Typography>
                <Typography variant="h6">Are there any Microtransactions on
Fotorama?</Typography>
                <Typography >
                  Fotorama is free, and will always be. It doesn't have any kinds of
Microtransactions.
                </Typography>
                <Typography variant="h6">How to connect with people?</Typography>
                <Typography >
                  Do you see the Posts of different users? Do you like them? Then you can
befriend them with a simple click on their Profile Picture that leads you to their
Profile.
                </Typography>
                <Typography variant="h6">My question is not listed above. What do I
do?</Typography>
                <Typography >
                  Don't worry. Write us at help(at)fotorama(dot)com. We'll definitely ping
you within a day regarding your query.
                </Typography>
              </DialogContent>
            </Dialog>
        <Dialog className='dialog'
          aria-describedby="alert-dialog-description" onClose={this.handlePolicyModal}
open={this.state.openPolicy}>

          <DialogContent>
            <div className='paper'>
              <h2 id="transition-modal-title">Fotorama Privacy Policy</h2>

              <span dangerouslySetInnerHTML={template} />
            </div>
          </DialogContent>
        </Dialog>
      </Grid>
    )
   }
```

```
}

export default withRouter(Footer);
```

**GuestLinks/index.js:**

```
import { Button } from "@material-ui/core"
import { Component } from "react"
import firebase from 'firebase/app'
import { withRouter } from "react-router";

class GuestLinks extends Component {

   constructor(props){
       super(props)
       this.state = {
           isAuthenticated: false,
           user: null
       }
   }

   render() {
       return (

           <div className='sectionDesktop navButtons'>
               <Button component="a" href='/' >
                   Home
               </Button>
               <Button component="a" href='/ourstory'>
                   Our Story
               </Button>
           </div>
       )
   }

}
export default withRouter(GuestLinks);
```

**Navbar/index.js:**
```
import React, { Component } from 'react'
import { AppBar, Toolbar, IconButton, Typography, Drawer,InputBase } from
'@material-ui/core'
import useScrollTrigger from '@material-ui/core/useScrollTrigger'
import Logo from '../../logo.png'
import { Menu,Search } from '@material-ui/icons'
import firebase from 'firebase/app'
import { withRouter } from "react-router"
import GuestLinks from '../GuestLinks'
import SignInLinks from '../SignInLinks'

function ElevationScroll(props) {
  const { children, window } = props;
  const trigger = useScrollTrigger({
    disableHysteresis: true,
    threshold: 0,
    target: window ? window() : undefined,
  });

  return React.cloneElement(children, {
    elevation: trigger ? 4 : 0,
  });
}


class Navbar extends Component {

  constructor(props) {
    super(props)
    this.state = {
      isAuthenticated: false,
      user: null,
      isLoading: true,
      search: '',
    }
  }

   componentDidMount()   {
    firebase.auth().onAuthStateChanged((user)=>{
      if (user) {
        this.setState({
          isAuthenticated: true,
          user: user,
          isLoading: false,
        })
      } else {
        this.setState({
          isAuthenticated: false,
          user: null,
          isLoading: false,
        })
      }
    })

  }


  handleSearchVal = (e) => {
    this.setState({
      search: e.target.value
    })
  }
  handleSearch = (e) => {
    if(this.state.search.length===0) return
    if (e.key === 'Enter') {
      this.props.history.push('/search/' + this.state.search)
      return
    }
  }
  toggleDrawer = () => {
    this.setState({
      ...this.state,
      leftDrawer: !this.state.leftDrawer
    })
  };

  handlePolicyModal = (event, reason) => {
    if (reason === 'clickaway') {
      return
    }
    this.setState({
      ...this.state,
      openPolicy: !this.state.openPolicy,
    })
  }
  handleHelpModal = (event, reason) => {
    if (reason === 'clickaway') {
      return
    }
    this.setState({
      ...this.state,
      openHelp: !this.state.openHelp,
    })
  }




  render() {
    return (
      <div className='grow'>
        <ElevationScroll props={this.props}>
          <AppBar position="fixed">
            <Toolbar>
              {this.state.isAuthenticated ?
                <div>
                  <IconButton
                    edge="start"
                    className='menuButton'
                    onClick={this.toggleDrawer}
                    color="inherit"
                    aria-label="open drawer"
                  >
                    <Menu />
                  </IconButton>
                  <React.Fragment key="left">
                    <Drawer anchor="left" open={this.state.leftDrawer}
onClose={this.toggleDrawer}>
                      <SignInLinks toggleDarkMode={this.props.toggleDarkMode}/>
                    </Drawer>
                  </React.Fragment>
                </div>
                : null}
              <img src={Logo} alt="Logo" className='iconDesktop' />
              <Typography href="/" component="a" className='title iconDesktop'
variant="h6" noWrap>
                Fotorama
              </Typography>

              <div className='grow' />

          <div className='search'>
            <div className='searchIcon'>
              <Search />
            </div>
            <InputBase
              placeholder="Search Fotorama…"
              className='inputBase'
              inputProps={{ 'aria-label': 'search' }}
              onKeyUp={this.handleSearch}
              onChange={this.handleSearchVal}
            />
          </div>

              {
              this.state.isLoading ? null :
              !this.state.isAuthenticated?
                <GuestLinks />:null
              }
            </Toolbar>
          </AppBar>
        </ElevationScroll>
      </div>

    )
  }

}

export default withRouter(Navbar);
```

**Policy/index.js:**

```
import { Skeleton } from "@material-ui/lab"
import { Component } from "react"
import { Card, CardActions, CardHeader, Avatar, IconButton, CardContent, Dialog, Paper,
CircularProgress, Tooltip, Zoom, Box, Menu, MenuItem, Link, Typography } from
'@material-ui/core'
import { MoreVert, Favorite, Delete, GetApp } from '@material-ui/icons'
import { withRouter } from 'react-router-dom'
import firebase from 'firebase'

class Post extends Component {
 _isMounted = false;
 constructor(props) {
   super(props)
   this.state = {
     postID: this.props.id,
     byID: '',
     byAvatar: '',
     byDisplayName: '',
     isLoading: true,
     imageUrl: '',
     user: null,
     postContent: '',
     likes: [],
     timestamp: '',
     downloads: 0,
     isLiked: false,
     tags: [],
     deleted: false,
     isProfilePicModalOpen: false,
     isPhotoModalOpen: false,
     isLiking: false,
     isDownloading: false,
     menuAnchorEl: null,
   }
 }

 async componentDidMount() {
   this._isMounted = true
   firebase.auth().onAuthStateChanged((user) => {
     if (user) {
       this.setState({
         user: user
       })
     } else {
       this.props.history.push('/')
```

```
    }
  })

  const dbRef = firebase.firestore()

  const postRef = dbRef.collection('posts').doc(this.state.postID)

  const result = await postRef.get()

  if (!result.exists) {
    this.setState({
      deleted: true
    })
    return
  }
  const data = await result.data()
  await this.setState({
    byID: data.byID,
    imageUrl: data.imageUrl,
    postContent: data.postContent,
    likes: data.likes,
    timestamp: data.timestamp,
    downloads: data.downloads,
    tags: data.tags,
  })
  if (this.state.likes.includes(this.state.user.uid)) {
    this.setState({
      isLiked: true
    })
  }
  const userRef = dbRef.collection('users').doc(this.state.byID)

  const userResult = await userRef.get()
  if (!userResult.exists) {
    this.setState({
      deleted: true
    })
    return
  }
  const userData = await userResult.data()
  this.setState({
    byDisplayName: userData.displayName,
    byAvatar: userData.photoURL,
    isLoading: false,
  })
```

```
}


toggleLike = async () => {
  this.setState({
    isLiking: true,
  })
  const dbRef = firebase.firestore()
  const postRef = dbRef.collection('posts').doc(this.state.postID)
  const postResult = await postRef.get()
  if (!postResult.exists) {
    this.setState({
      deleted: true,
    })
    return
  }
  const postData = await postResult.data()


  if (this.state.isLiked) {
    await postRef.set({
      likes: postData.likes.filter((e) => e !== this.state.user.uid)
    }, { merge: true })

    this.setState({
      ...this.state,
      likes: postData.likes.filter((e) => e !== this.state.user.uid)
    })

    this.setState({
      isLiked: false
    })
  } else {
    await postRef.set({
      likes: [...postData.likes, this.state.user.uid]
    }, { merge: true })

    this.setState({
      ...this.state,
      likes: [...postData.likes, this.state.user.uid]
    })

    await Promise.all(this.state.tags.map(async tag => {
      const userTagRef =
dbRef.collection('users').doc(this.state.user.uid).collection('likedTags').doc(tag)
      const userTagResult = await userTagRef.get()
      const userTagData = await userTagResult.data()
      if (!userTagResult.exists) {
        await userTagRef.set({
          value: 1
        })
        return
      }
      await userTagRef.set({
        value: userTagData.value + 1
      })
    }))

    this.setState({
      isLiked: true
    })

  } this.setState({
    isLiking: false,
  })
}

downloadImage = async () => {
  this.setState({
    isDownloading: true
  })
  const dbRef = firebase.firestore()
  const postRef = dbRef.collection('posts').doc(this.state.postID)
  const postResult = await postRef.get()
  if (!postResult.exists) {
    this.props.history.push('/')
    return
  }
  const postData = await postResult.data()

  await postRef.set({
    downloads: postData.downloads + 1
  }, { merge: true })

  var url = this.state.imageUrl.replace(/^data:image\/[^;]+/,
'data:application/octet-stream')
  window.open(url)

  this.setState({
    isDownloading: false
  })
}
```

```
togglePhotoModal = () => {
  this.setState({
    ...this.state,
    isPhotoModalOpen: !this.state.isPhotoModalOpen,
  })
}

toggleProfilePicModal = () => {
  this.setState({
    ...this.state,
    isProfilePicModalOpen: !this.state.isProfilePicModalOpen,
  })
}

handleMenuOpen = (e) => {
  this.setState({
    ...this.state,
    menuAnchorEl: e.currentTarget
  })
}

handleMenuClose = () => {
  this.setState({
    ...this.state,
    menuAnchorEl: null
  })
}

getTime = (timestamp) => {
  if (this.state.isLoading) return ''
  let d = timestamp
  let currentTime = new Date().getTime()

  const monthNames = ["January", "February", "March", "April", "May", "June",
    "July", "August", "September", "October", "November", "December"
  ];
  const min = 60 * 1000
  const hour = 60 * min
  if (currentTime - d <= 15 * min) return 'Few mins ago'
  else if (currentTime - d <= 30 * min) return 'Half hour ago'
  else if (currentTime - d <= hour) return 'An hour ago'
  else if (currentTime - d <= 2 * hour) return '2 hours ago'
  else if (currentTime - d <= 4 * hour) return '4 hours ago'
  else if (currentTime - d <= 8 * hour) return '8 hours ago'
  else if (currentTime - d <= 12 * hour) return '12 hours ago'
```

```
   else if (currentTime - d <= 24 * hour) return 'A day ago'

   d = new Date(d)
   return monthNames[d.getMonth()] + ' ' + d.getDate() + ', ' + d.getFullYear()

 }

 render() {
   if (this.state.isLoading)
     return <Skeleton type='rect' className='postPlaceholder' />
   if (this.state.deleted) {
     return (<Paper elevation={0} className='postPaper'>
       <Card variant='outlined' className='createPostCard'>
         <CardContent>
           This post has been deleted by the user or the user closed his/her Fotorama
Account or has been taken down due to violation of one or more Polcies under Fotorama
Healthy Post Guidelines.
         </CardContent>
       </Card>
     </Paper>)
   }
   return (
     <Paper elevation={0} className='postPaper'>
       <Card variant='outlined' className='createPostCard'>
         <CardHeader
           avatar={
             <Tooltip TransitionComponent={Zoom} title="Open Avatar" aria-label="Open
Avatar" arrow>
               <Avatar alt={this.state.byDisplayName}
src={this.state.byAvatar.replace('s96-c', 's500-c')} onClick={this.toggleProfilePicModal}
/>
             </Tooltip>
           }
           action={
             this.state.user.uid === this.state.byID ?
               <IconButton aria-label="settings" onClick={this.handleMenuOpen}>
                 <MoreVert />
               </IconButton>
               :
               null
           }
           title={
             <Tooltip TransitionComponent={Zoom} title="Open Profile" aria-label="Open
Profile" arrow>
               <Link color='textPrimary' href={'/profile/' + this.state.byID}>
                 {this.state.byDisplayName}
```

```
              </Link>
            </Tooltip>
          }
          subheader={this.getTime(this.state.timestamp)}
        />
        <CardContent>
          {this.state.postContent.split('\n').map((e) => { return <p>{e}</p> })}
        </CardContent>
        {
          this.state.imageUrl.length !== 0
            ?
            <div>
              <img alt='Upload'
                className='createPostCardImage'
                src={this.state.imageUrl} onClick={this.togglePhotoModal}
              />
            </div>

            :
            null

        }
        <CardActions disableSpacing>
          {
            this.state.isLiking
              ?
              <CircularProgress color='secondary' />
              :
              <Box display='flex' alignItems='center'>
                <Tooltip TransitionComponent={Zoom} title="Like Post" aria-label="Like
Photo" arrow>
                  <IconButton aria-label="Like Post" onClick={this.toggleLike}
color={this.state.isLiked ? 'secondary' : 'default'}>
                    <Favorite />
                  </IconButton>
                </Tooltip>
                {this.state.likes.length}
              </Box>
          }
          {
            this.state.isDownloading
              ?
              <CircularProgress color='secondary' />
              :
              <Box display='flex' alignItems='center' style={{ marginLeft: 20 }}>
```

```
                 <Tooltip TransitionComponent={Zoom} title="Download Image"
aria-label="Download Image" arrow>
                   <IconButton aria-label="Download Image" onClick={this.downloadImage} >
                     <GetApp />
                   </IconButton>
                 </Tooltip>
                 {this.state.downloads}
               </Box>
           }
         </CardActions>
       </Card>

       <Dialog className='dialog'
         aria-labelledby='Profile Photo Dialog'
         aria-describedby='Profile Photo' onClose={this.toggleProfilePicModal}
open={this.state.isProfilePicModalOpen}>
         <img src={this.state.byAvatar.replace('s96-c', 's500-c')} alt='Profile' />
       </Dialog>
       <Dialog className='dialog'
         aria-labelledby='Photo Dialog'
         aria-describedby='Photo' onClose={this.togglePhotoModal}
open={this.state.isPhotoModalOpen}>
         <img src={this.state.imageUrl} alt='Upload' />
       </Dialog>
       <Menu
         anchorEl={this.state.menuAnchorEl}
         keepMounted
         open={Boolean(this.state.menuAnchorEl)}
         onClose={this.handleMenuClose}
       >
         <MenuItem onClick={() => { this.props.deletePost(this.state.postID) }}><Delete
/> &nbsp;Delete Post</MenuItem>
       </Menu>
     </Paper>
   )
 }

}

export default withRouter(Post);
```

**SignInLinks/index.js:**

```
import { Avatar, Dialog, DialogContent, DialogTitle, Divider, List, ListItem,
ListItemIcon, ListItemText, Typography,Box,IconButton } from "@material-ui/core"
import { AccountCircle, ExitToApp, GroupWork, Help, Home,Brightness4 } from
"@material-ui/icons"
import { Component } from "react"
import { withRouter } from 'react-router-dom'
import firebase from 'firebase/app'
import policy from '../policy'
var template = { __html: policy };

class SignInLinks extends Component {

  constructor(props) {
   super(props)
   this.state = {
     openPolicy: false,
     openHelp: false,
     isAuthenticated: false,
     user: null,
     isLoading: true,
     isModalOpen: false,
   }

 }

 componentDidMount()   {
   firebase.auth().onAuthStateChanged((user)=>{
     if (user) {
       user.photoURL.replace('s96-c','s500-c')
       this.setState({
         isAuthenticated: true,
         user: user,
         isLoading: false,
       })
     } else {
       this.props.history.push('/')
     }
   })

 }


 preventDefault = (event) => event.preventDefault();
 handlePolicyModal = (event, reason) => {
   if (reason === 'clickaway') {
     return
   }
   this.setState({
     ...this.state,
     openPolicy: !this.state.openPolicy,
   })
 }
 handleHelpModal = (event, reason) => {
   if (reason === 'clickaway') {
     return
   }
   this.setState({
     ...this.state,
     openHelp: !this.state.openHelp,
   })
 }

 signOutMethod = async () => {
   await firebase.auth().signOut()
   this.props.history.push('/')
 }


 toggleModalOpen = () => {
   this.setState({
     ...this.state,
     isModalOpen: !this.state.isModalOpen
   })
 }
 render() {
   if(this.state.isLoading) return (null);
   return (
     <div className='list'>
       <List>
         <ListItem style={{display:'block'}}>
           <Box display='flex' alignItems='center' justifyContent='space-between'>

           <Avatar alt={this.state.user.displayName}
src={this.state.user.photoURL.replace('s96-c','s500-c')} className='drawerPhoto'
onClick={this.toggleModalOpen} />
           <IconButton onClick={()=>{this.props.toggleDarkMode()}}>
             <Brightness4/>
           </IconButton>
           </Box>
           <Typography variant="h6" >Hey There,<br/>
{this.state.user.displayName}</Typography>
```

```
         </ListItem>
         <Divider />
         <ListItem button key="Home" component="a" href="/">
           <ListItemIcon> <Home /></ListItemIcon>
           <ListItemText primary="Home" />
         </ListItem>
         <Divider />
         <ListItem button key="Your Profile" component="a"
href={"/profile/"+this.state.user.uid}>
           <ListItemIcon> <AccountCircle /></ListItemIcon>
           <ListItemText primary="Your Profile" />
         </ListItem>
         <Divider />
         <ListItem button key="Our Story" component="a" href="/ourstory/">
           <ListItemIcon> <GroupWork /></ListItemIcon>
           <ListItemText primary="Our Story" />
         </ListItem>
         <ListItem button key="Help" onClick={this.handleHelpModal}>
           <ListItemIcon> <Help /></ListItemIcon>
           <ListItemText primary="Help" />
         </ListItem>
         <ListItem button key="Sign Out" onClick={this.signOutMethod}>
           <ListItemIcon> <ExitToApp /></ListItemIcon>
           <ListItemText primary="Sign Out" />
         </ListItem>
         <Divider />
         <ListItem key="Copyright">
           <ListItemText secondary="© Copyright 2021 - Present" />
         </ListItem>
         <div>
           <ListItem onClick={this.handlePolicyModal} component="a" href="#" key="Privacy
Policy">
             <ListItemText secondary="Privacy Policy | Fotorama" />
           </ListItem>
           <Dialog className='dialog'
             aria-labelledby="alert-dialog-title"
             aria-describedby="alert-dialog-description" onClick={this.handleHelpModal}
open={this.state.openHelp}>

             <DialogTitle id="alert-dialog-title">Fotorama | Help & FAQ</DialogTitle>
             <DialogContent>
               <Typography variant="h6">What is Fotorama all about?</Typography>
               <Typography >
                 Fotorama is what you all think it is! Snap your Pictures, Share them
with your Loved Ones, friends and families! Cherish your memories!
               </Typography>
               <Typography variant="h6">What Information do you store about
me?</Typography>
               <Typography >
                 Apart from the content that you post on Fotorama, nothing else.
               </Typography>
               <Typography variant="h6">What more can I do after logging in?</Typography>
               <Typography >
                 You can share your snaps, make friends, search posts based on Tags and
more!
               </Typography>
               <Typography variant="h6">How do I post?</Typography>
               <Typography >
                 Do you see that Share button on your Dashboard? Click it and find the
rest of the steps!
               </Typography>
               <Typography variant="h6">Are there any Microtransactions on
Fotorama?</Typography>
               <Typography >
                 Fotorama is free, and will always be. It doesn't have any kinds of
Microtransactions.
               </Typography>
               <Typography variant="h6">How to connect with people?</Typography>
               <Typography >
                 Do you see the Posts of different users? Do you like them? Then you can
befriend them with a simple click on their Profile Picture that leads you to their
Profile.
               </Typography>
               <Typography variant="h6">My question is not listed above. What do I
do?</Typography>
               <Typography >
                 Don't worry. Write us at help(at)fotorama(dot)com. We'll definitely ping
you within a day regarding your query.
               </Typography>
             </DialogContent>
           </Dialog>

           <Dialog className='dialog'
             aria-labelledby='Profile Photo Dialog'
             aria-describedby='Profile Photo' onClose={this.toggleModalOpen}
open={this.state.isModalOpen}>
               <img src={this.state.user.photoURL.replace('s96-c','s500-c')}
alt='Profile'/>
           </Dialog>
           <Dialog className='dialog'
             aria-describedby="alert-dialog-description" onClose={this.handlePolicyModal}
open={this.state.openPolicy}>
```

```
             <DialogContent>
               <div className='paper'>
                 <h2 id="transition-modal-title">Fotorama Privacy Policy</h2>

                 <span dangerouslySetInnerHTML={template} />
               </div>
             </DialogContent>
           </Dialog>

         </div>
       </List>

     </div>
   )
 }

}

export default withRouter(SignInLinks);
```

**UserView/index.js:**

```
import { Component } from "react";
import { withRouter } from 'react-router-dom'
import firebase from 'firebase'
import { Box, Avatar, Typography,Dialog,Card,Link} from '@material-ui/core'
import { Skeleton } from '@material-ui/lab'

class UserView extends Component {

 constructor(props) {
   super(props)
   this.state = {
     id: this.props.id,
     displayName: '',
     photoURL: '',
     followers: 0,
     isLoading: true,
     isProfilePicModalOpen:false,
   }
 }

 async componentDidMount() {
   const dbRef = firebase.firestore()

   const userRef = dbRef.collection('users').doc(this.state.id)
   const followerRef =
dbRef.collection('users').doc(this.state.id).collection('followers')

   const userResult = await userRef.get()
   const followerResult = await followerRef.get()

   if (!userResult.exists) {
     this.props.history.push('/')
     return
   }

   const userData = userResult.data()
   const followerData = followerResult.docs

   this.setState({
     ...this.state,
     displayName: userData.displayName,
     photoURL: userData.photoURL,
     followers: followerData.length,
     isLoading: false,
   })
 }


 toggleProfilePicModal = () => {
   this.setState({
     ...this.state,
     isProfilePicModalOpen: !this.state.isProfilePicModalOpen
   })
 }

 render() {
   if (this.state.isLoading)
     return (
       <Skeleton type='rect' className='userViewPlaceholder' />
     )
   return (
     <Link href={'/profile/'+this.state.id} className='userViewLink'>
     <Card variant='outlined' style={{marginTop: 10,marginBottom: 10}}>
     <Box display='flex' justifyContent='space-between' alignItems='center'
style={{padding: 10}}>
         <Box display='flex' alignItems = 'center'>
         <Avatar alt={this.state.displayName} src={this.state.photoURL.replace('s96-c',
's500-c')}  onClick={this.toggleProfilePicModal} />
         <Typography style={{marginLeft: 10}}>{this.state.displayName}</Typography>
         </Box>
         <Typography>{this.state.followers}
{this.state.followers===1?'Follower':'Followers'}</Typography>

       <Dialog
         aria-labelledby='Profile Photo Dialog'
         aria-describedby='Profile Photo' onClose={this.toggleProfilePicModal}
open={this.state.isProfilePicModalOpen}>
         <img src={this.state.photoURL.replace('s96-c', 's500-c')} alt='Profile' />
       </Dialog>
     </Box>
     </Card>
     </Link>
   )
 }

}

export default withRouter(UserView);
```

**Dashboard/index.js:**

```
import { Component } from "react"
import { withRouter } from 'react-router-dom'
import { Grid} from '@material-ui/core'
import firebase from 'firebase'
import { Skeleton } from '@material-ui/lab'
import CreatePost from '../../components/CreatePost'
import Post from '../../components/Post'
import InfiniteScroll from 'react-infinite-scroll-component'
import index from '../../config/algoliaConfig'


class Dashboard extends Component {

 constructor(props) {
   super(props)
   this.state = {
     user: null,
     isAuthenticated: false,
     posts: [],
     isUserDataLoading: true,
     isPostsLoading: true,
     lastTimeStamp: new Date().getTime(),
     hasMore: true,

   }
 }
 async componentDidMount() {

   firebase.auth().onAuthStateChanged(async (user) => {
     if (user) {

       const db = firebase.firestore()
       const ref = db.collection('users').doc(user.uid)
       const result = await ref.get()
       if (!result.exists) {
         await index.saveObjects([{objectID:user.uid,displayName:'Krunal
Rank',uid:user.uid}])
         await ref.set({
           displayName: user.displayName,
           photoURL: user.photoURL,
           followers: [],
           following: [],
         })
       }
       this.setState({
         user: user,
         isUserDataLoading: false,
         isAuthenticated: true
       })
     } else {
       this.props.history.push('/')
     }
     return
   })


   const dbRef = firebase.firestore()

   const postRef = dbRef.collection('posts').orderBy('timestamp', 'desc').limit(1)
   const postResult = await postRef.get()
   const postRefs = await postResult.docs

   let posts = []
   let ts = 0
   await Promise.all(postRefs.map(async (e) => {
     const id = await e.id
     posts = [...posts, id]
     ts = e.data().timestamp
   }))
   this.setState({
     ...this.state,
     posts: posts,
     lastTimeStamp: ts,
     isPostsLoading: false,
   })
 }

 loadNewPosts = async () => {
   await this.setState({
     lastTimeStamp: new Date().getTime(),
     posts: [],
   })
   const db = firebase.firestore()
   const postRef = db.collection('posts').where('timestamp', '<',
this.state.lastTimeStamp).orderBy('timestamp','desc').limit(10)
   const postResult = await postRef.get()
   const postRefs = await postResult.docs

   if(postRefs.length===0){
     await this.setState({
       hasMore: false,
     })
     return
   }
   let posts = []
   let ts = 0
   await Promise.all(postRefs.map(async (e) => {
     const id = await e.id
     posts = [...posts, id]
     ts = e.data().timestamp
   }))

   this.setState({
     ...this.state,
     posts: posts,
     lastTimeStamp: ts
   })
 }
 loadMorePosts = async () => {
   if(this.state.isPostsLoading) return
   const db = firebase.firestore()
   const postRef = db.collection('posts').where('timestamp', '<',
this.state.lastTimeStamp).orderBy('timestamp','desc').limit(10)
   const postResult = await postRef.get()
   const postRefs = await postResult.docs

   let posts = []
   let ts = 0
   await Promise.all(postRefs.map(async (e) => {
     const id = await e.id
     posts = [...posts, id]
     ts = e.data().timestamp
   }))

   this.setState({
     ...this.state,
     posts: [...this.state.posts, ...posts],
     lastTimeStamp: ts
   })

 }

 deletePost = async (id) => {
   if (this.state.isPostsLoading) return
   const dbRef = firebase.firestore()
   const postRef = dbRef.collection('posts').doc(id)
   const postResult = await postRef.get()
   const postData = postResult.data()
   this.setState({
     ...this.state,
     posts: this.state.posts.filter((e) => e !== id)
   })

   await Promise.all(postData.tags.map(async (tag) => {
     const tagRef = dbRef.collection('tags').doc(tag)
     const tagResult = await tagRef.get()
     const tagData = await tagResult.data()
     await tagRef.set({
       posts: tagData.posts.filter((e) => e !== id)
     })
   }))
   await postRef.delete()
 }

 render() {
   return (
     <Grid className='dashboard'>
       <Grid item xs={12}>
         {
           this.state.isUserDataLoading ?
             <Skeleton variant="rect" className='dashboardCreatePostPlaceholder' />
             :
             <CreatePost user={this.state.user} />
         }
       </Grid>
       <InfiniteScroll
         dataLength={this.state.posts.length}
         next={this.loadMorePosts}
         hasMore={true}
         refreshFunction={this.loadNewPosts}
         pullDownToRefresh
         pullDownToRefreshThreshold={50}
         pullDownToRefreshContent={
           <h3 style={{ textAlign: 'center' }}>&#8595; Pull down to refresh</h3>
         }
         releaseToRefreshContent={
           <h3 style={{ textAlign: 'center' }}>&#8593; Release to refresh</h3>
         }
       >
         {this.state.posts.map((id) => {
           return (
             <Grid item xs={12}>
               <Post key={id} deletePost={this.deletePost} id={id} />
```

```
                </Grid>
              )
            })}
          </InfiniteScroll>
        </Grid>
      )
    }

}

export default withRouter(Dashboard);
```

**Home/index.js:**

```
import { Component } from "react"
import { Container, Button, Typography } from '@material-ui/core'
import Logo from '../../logo.png'
import GoogleLogo from '../../assets/google_logo.png'
import firebase from 'firebase'
import Skeleton from '@material-ui/lab/Skeleton'
import { withRouter } from 'react-router-dom'
import Dashboard from '../Dashboard'
class Home extends Component {

 constructor(props) {
   super(props)
   this.state = {
     isAuthenticated: false,
     user: false,
     isLoading: true,
   }
 }

 componentDidMount() {
   firebase.auth().onAuthStateChanged((user) => {
     if (user) {
       this.setState({
         isAuthenticated: true,
         user: user,
         isLoading: false
       })
     } else {
       this.setState({
         isAuthenticated: false,
         user: null,
         isLoading: false,
       })
     }
   })
 }


 signInMethod = async () => {
   const provider = new firebase.auth.GoogleAuthProvider()
   provider.addScope("profile")
   try {
     await firebase.auth().signInWithPopup(provider);
     const user = await firebase.auth().currentUser;
     if (!user) return;
```

```
      this.setState({
        isAuthenticated: true,
        user: user,
        isLoading: false
      })

    } catch (err) {
      console.log(err);

    }

  }


  render() {

    if (this.state.isAuthenticated) return (<Dashboard />)
    if (this.state.isLoading)
      return (
        <Container className='home'>
          <Container className='homeCenter'>
            <Skeleton variant="circle" className='homeLogoPlaceholder' />
          </Container>
          <Container className='homeCenter'>
            <Skeleton variant="rect" className='homeTitlePlaceholder' />
          </Container>
          <Container className='homeCenter'>
            <Skeleton variant="rect" className='homeButtonPlaceholder' />
          </Container>
        </Container>
      )
    return (
      <Container className='home'>
        <Container className='homeCenter'>
          <img src={Logo} alt="Logo" className='homeLogo' />
        </Container>
        <Container className='homeCenter'>
          <Typography variant='h3'>
            Fotorama
          </Typography>
        </Container>
        <Container className='homeCenter'>
          <Button onClick={this.signInMethod} className='signInButton' variant="contained"
color="primary">
            <img src={GoogleLogo} alt="GoogleLogo" className='homeGoogleLogo' />
            SIGN IN WITH GOOGLE
          </Button>
        </Container>
      </Container>
    )
  }

}

export default withRouter(Home);
```

**OurStory/index.js:**

```
import React from "react";
import Button from "@material-ui/core/Button";
import Divider from "@material-ui/core/Divider";
import Container from "@material-ui/core/Container";
import Grid from "@material-ui/core/Grid";
import { Typography,Card } from "@material-ui/core";
import Logo from "../../logo.png";

import { withRouter } from 'react-router-dom'

class AboutUs extends React.Component {

 constructor(props) {
   super(props)
   this.state = {
     leftDrawer: false,
   }
 }
 render() {
   return (
     <Container className='home'>
     <Card  variant='outlined' style={{padding:10}}>
       <Grid container >
         <Grid item xs={12} className='homeCenter'>
           <div className=''>
             <img alt="XPert Logo" className='homeLogo' src={Logo} />
             <div style={{ fontSize: 30,textAlign: 'center' }}>Fotorama</div>
           </div>
         </Grid>
         <Grid container xs={12} justify="start">
           <Grid item>
             <Typography variant="h6" align="left">The Company</Typography>
             <p>
               Fotorama is an initiative to provide people throughout the world an
               opportunity to share their experiences in the most independent way
possible.
               The Posts available on the website are completely copyrighted by their
respective owners, as
               per the agreement in our Privacy Policy. For more details, please refer
the Privacy Policy | Fotorama.

               </p>
             <p align="left">
               Fotorama started as a Social Media App in March, 2021 and is running
since, with the love
```

```
               and wishes of the users.
               </p>
             <p align="left">
               It started as a Project and minimal idea to allow users to reach each
other via their Posts
               but it has continuously grown since then.
               </p>
           </Grid>
           <Grid item xs={12}>
             <Typography variant="h6" align="left">The Team</Typography>
             <p align="left">
               Krunal Rank, Founder and CEO, Fotorama
               </p>
           </Grid>
           <Grid item xs={12}>
             <Typography variant="h6" align="left">Contact</Typography>
             <p align="left">
               Email: hola(at)fotorama(dot)com
               </p>
             <p align="left">
               Phone: +91 701 650 7648
               </p>
             <p align="left">
               Fax: +22 675 124 5
               </p>
             <p align="left" color="textSecondary">
               Fotorama<br />
                   NIT Surat, Surat<br />
                   India<br />
                   395007
               </p>
           </Grid>
         </Grid>

       </Grid>
     </Card>
     </Container>
   );
 }
}

export default withRouter(AboutUs);
```

**Profile/index.js:**

```
import { Component } from "react"
import { withRouter } from 'react-router-dom'
import { Grid, Paper, Typography, Avatar, Dialog, Box, Button, CircularProgress, Tabs,
Tab, Card, Tooltip, Zoom } from '@material-ui/core'
import { ExitToApp, PersonAdd, PersonAddDisabled } from '@material-ui/icons'
import firebase from 'firebase'
import { Skeleton } from '@material-ui/lab'
import Post from '../../components/Post'
import UserView from '../../components/UserView'
import InfiniteScroll from 'react-infinite-scroll-component'

class Profile extends Component {

 constructor(props) {
   super(props)
   this.state = {
     id: this.props.match.params.id,
     displayName: '',
     photoURL: '',
     followers: [],
     following: [],
     isFollower: true,
     isFollowing: false,
     isLoading: true,
     posts: [],
     isProfilePicModalOpen: false,
     currentUser: null,
     isFollowButtonClicked: false,
     lastTimeStamp: new Date().getTime(),
     tabValue: 0,
     isPostsLoading: true,
   }
 }

 async componentDidMount() {

   firebase.auth().onAuthStateChanged((user) => {
     if (user) {
       this.setState({
         currentUser: user
       })
     } else {
       this.props.history.push('/')
     }
   })
   const dbRef = firebase.firestore()

   const userRef = dbRef.collection('users').doc(this.state.id)
   const userResult = await userRef.get()
   if (!userResult.exists) {
     this.props.history.push('/')
     return
   }

   const userData = userResult.data()

   this.setState({
     ...this.state,
     displayName: userData.displayName,
     photoURL: userData.photoURL,
   })

   const currentUserID = this.state.currentUser.uid

   if (currentUserID !== this.state.id) {

     const followerRef =
dbRef.collection('users').doc(currentUserID).collection('followers').doc(this.state.id)
     const followingRef =
dbRef.collection('users').doc(currentUserID).collection('following').doc(this.state.id)

     const followerResult = await followerRef.get()
     const followingResult = await followingRef.get()

     if (followerResult.exists) {
       this.setState({
         ...this.state,
         isFollower: true
       })
     }
     if (followingResult.exists) {
       this.setState({
         ...this.state,
         isFollowing: true,
       })
     }
   }

   const followerResult = await
dbRef.collection('users').doc(this.state.id).collection('followers').get()
```

```
   let followers = []

   followerResult.docs.forEach((doc) => {
     followers = [...followers, doc.id]
   })
   await this.setState({
     followers: followers
   })

   const followingResult = await
dbRef.collection('users').doc(this.state.id).collection('following').get()

   let following = []

   followingResult.docs.forEach((doc) => {
     following = [...following, doc.id]
   })
   await this.setState({
     following: following
   })

   await this.setState({
     ...this.state,
     isLoading: false,
   })


   const postRef = dbRef.collection('posts').where('byID', '==',
this.state.id).where('timestamp', '<', this.state.lastTimeStamp).orderBy('timestamp',
'desc').limit(1)
   const postResult = await postRef.get()
   const postRefs = await postResult.docs

   let posts = []
   let ts = 0
   await Promise.all(postRefs.map(async (e) => {
     const id = await e.id
     posts = [...posts, id]
     ts = e.data().timestamp
   }))
   await this.setState({
     ...this.state,
     posts: posts,
     lastTimeStamp: ts,
     isPostsLoading: false
   })
```

```
 }


 loadNewPosts = async () => {
   await this.setState({
     lastTimeStamp: new Date().getTime(),
     posts: [],
   })
   const db = firebase.firestore()
   const postRef = db.collection('posts').where('byID', '==',
this.state.id).where('timestamp', '<', this.state.lastTimeStamp).orderBy('timestamp',
'desc').limit(10)
   const postResult = await postRef.get()
   const postRefs = await postResult.docs

   if (postRefs.length === 0) {
     await this.setState({
       hasMore: false,
     })
     return
   }
   let posts = []
   let ts = 0
   await Promise.all(postRefs.map(async (e) => {
     const id = await e.id
     posts = [...posts, id]
     ts = e.data().timestamp
   }))

   this.setState({
     ...this.state,
     posts: posts,
     lastTimeStamp: ts
   })
 }

 loadMorePosts = async () => {
   const db = firebase.firestore()
   const postRef = db.collection('posts').where('byID', '==',
this.state.id).where('timestamp', '<', this.state.lastTimeStamp).orderBy('timestamp',
'desc').limit(10)
   const postResult = await postRef.get()
   const postRefs = await postResult.docs

   let posts = []
   let ts = 0
   await Promise.all(postRefs.map(async (e) => {
     const id = await e.id
     posts = [...posts, id]
     ts = e.data().timestamp
   }))

   this.setState({
     ...this.state,
     posts: [...this.state.posts, ...posts],
     lastTimeStamp: ts
   })

 }

 deletePost = async (id) => {
   if (this.state.isPostsLoading) return
   const dbRef = firebase.firestore()
   const postRef = dbRef.collection('posts').doc(id)
   const postResult = await postRef.get()
   const postData = postResult.data()
   this.setState({
     ...this.state,
     posts: this.state.posts.filter((e) => e !== id)
   })

   await Promise.all(postData.tags.map(async (tag) => {
     const tagRef = dbRef.collection('tags').doc(tag)
     const tagResult = await tagRef.get()
     const tagData = await tagResult.data()
     await tagRef.set({
       posts: tagData.posts.filter((e) => e !== id)
     })
   }))
   await postRef.delete()
 }

 followPerson = async () => {
   this.setState({
     ...this.state,
     isFollowButtonClicked: true
   })

   const dbRef = firebase.firestore()
   const currentUserID = this.state.currentUser.uid
   const followingRef =
dbRef.collection('users').doc(currentUserID).collection('following').doc(this.state.id)
```

```
   const followerRef =
dbRef.collection('users').doc(this.state.id).collection('followers').doc(currentUserID)

   await followingRef.set({
     value: 1
   })
   await followerRef.set({
     value: 1
   })

   this.setState({
     ...this.state,
     isFollowing: true,
     followers: [...this.state.followers, this.state.currentUser.uid],
     isFollowButtonClicked: false
   })
 }


 unfollowPerson = async () => {
   this.setState({
     ...this.state,
     isFollowButtonClicked: true
   })

   const dbRef = firebase.firestore()
   const currentUserID = this.state.currentUser.uid
   const followingRef =
dbRef.collection('users').doc(currentUserID).collection('following').doc(this.state.id)
   const followerRef =
dbRef.collection('users').doc(this.state.id).collection('followers').doc(currentUserID)

   await followingRef.delete()
   await followerRef.delete()

   this.setState({
     ...this.state,
     isFollowing: false,
     followers: this.state.followers.filter((e) => e !== this.state.currentUser.uid),
     isFollowButtonClicked: false
   })
 }

 signOutMethod = async () => {
   await firebase.auth().signOut()
   this.props.history.push('/')
```

```
  }

  handleTabChange = (event, value) => {
    this.setState({
      ...this.state,
      tabValue: value
    })
  }

  toggleProfilePicModal = () => {
    this.setState({
      ...this.state,
      isProfilePicModalOpen: !this.state.isProfilePicModalOpen
    })
  }


  render() {
    if (this.state.isLoading)
      return (
        <Grid className='profile'>
          <Grid item xs={12}>
            <Skeleton type='rect' className='profilePlaceholder' />
          </Grid>
        </Grid>
      )
    return (
      <Grid className='profile'>
        <Grid item xs={12}>
          <Paper elevation={0}>
            <Grid className='profileTop'>
              <Box display='flex' justifyContent='center'>
                <Avatar alt={this.state.displayName}
src={this.state.photoURL.replace('s96-c', 's500-c')} className='profilePhoto'
onClick={this.toggleProfilePicModal} />
              </Box>
              <Box display='flex' justifyContent='center'>
                <Typography variant='h4'>{this.state.displayName}</Typography>
              </Box>
              {
                this.state.id === this.state.currentUser.uid
                  ?
                  <Box display='flex' justifyContent='center' alignItems='center'>
                    <Tooltip TransitionComponent={Zoom} title="Sign Out" aria-label="Sign
Out" arrow>
```

```
                     <Button onClick={this.signOutMethod} variant="contained"
color="secondary">
                       <ExitToApp />Sign Out
                     </Button>
                   </Tooltip>
                 </Box>
                 :
                 <Box display='flex' justifyContent='center' alignItems='center'>
                   {
                     this.state.isFollowButtonClicked
                       ?
                       <CircularProgress color='secondary' />
                       :
                       this.state.isFollowing
                         ?

                         <Tooltip TransitionComponent={Zoom} title="Follow Person"
aria-label="Unfollow Person" arrow>
                           <Button onClick={this.unfollowPerson} variant="contained"
color="secondary">
                             <PersonAddDisabled />Unfollow
                           </Button>
                         </Tooltip>
                         :
                         <Tooltip TransitionComponent={Zoom} title="Follow Person"
aria-label="Follow Person" arrow>
                           <Button onClick={this.followPerson} variant="contained"
color="secondary">
                             <PersonAdd />Follow
                           </Button>
                         </Tooltip>
                   }
                 </Box>
             }
             <Tabs
               value={this.state.tabValue}
               onChange={this.handleTabChange}
               indicatorColor="secondary"
               textColor="secondary"
               centered
             >
               <Tab label="Posts" />
               <Tab label="Followers" />
               <Tab label="Following" />
             </Tabs>
           </Grid>
```

```
         </Paper>
       </Grid>
       <Grid>
         {
           this.state.tabValue === 0
             ?
             this.state.isPostsLoading
               ?
               <Grid item xs={12} style={{ marginTop: 20 }}>
                 <Skeleton type='rect' className='profilePlaceholder' />
               </Grid>
               :
               this.state.posts.length === 0
                 ?
                 <Card variant='outlined' style={{ marginTop: 20 }}>
                   <Typography style={{ textAlign: 'center' }} variant='h4'> No
Posts</Typography>
                 </Card>
                 :
                 <InfiniteScroll
                   dataLength={this.state.posts.length}
                   next={this.loadMorePosts}
                   hasMore={true}
                   refreshFunction={this.loadNewPosts}
                   pullDownToRefresh
                   pullDownToRefreshThreshold={50}
                   pullDownToRefreshContent={
                     <h3 style={{ textAlign: 'center' }}>&#8595; Pull down to
refresh</h3>
                   }
                   releaseToRefreshContent={
                     <h3 style={{ textAlign: 'center' }}>&#8593; Release to refresh</h3>
                   }
                 >
                   {this.state.posts.map((id) => {
                     return (
                       <Grid item xs={12}>
                         <Post key={id} deletePost={this.deletePost} id={id} />
                       </Grid>
                     )
                   })}
                 </InfiniteScroll>
             :
             this.state.tabValue === 1
               ?
               this.state.isLoading
                 ?
                 <Grid item xs={12} style={{ marginTop: 20 }}>
                   <Skeleton type='rect' className='profilePlaceholder' />
                 </Grid>
                 :
                 this.state.followers.length === 0
                   ?
                   <Card variant='outlined' style={{ marginTop: 20 }}>
                     <Typography style={{ textAlign: 'center' }} variant='h4'> No
Followers</Typography>
                   </Card>
                   :
                   this.state.followers.map((follower) => {
                     return (
                       <Grid item xs={12}>
                         <UserView id={follower} />
                       </Grid>
                     )
                   })
               :
               this.state.isLoading
                 ?

                 <Grid item xs={12} style={{ marginTop: 20 }}>
                   <Skeleton type='rect' className='profilePlaceholder' />
                 </Grid>
                 :
                 this.state.following.length === 0
                   ?
                   <Card variant='outlined' style={{ marginTop: 20 }}>
                     <Typography style={{ textAlign: 'center' }} variant='h4'> No
Following</Typography>
                   </Card>
                   :
                   this.state.following.map((follower) => {
                     return (

                       <Grid item xs={12}>
                         <UserView id={follower} />
                       </Grid>
                     )
                   })
         }
       </Grid>
       <Dialog
         aria-labelledby='Profile Photo Dialog'
         aria-describedby='Profile Photo' onClose={this.toggleProfilePicModal}
open={this.state.isProfilePicModalOpen}>
         <img src={this.state.photoURL.replace('s96-c', 's500-c')} alt='Profile' />
       </Dialog>
     </Grid>
   )
 }

}

export default withRouter(Profile);
```

**SearchView/index.js:**

```
import { Component } from "react";

import { withRouter } from 'react-router-dom'
import { Grid, Paper, Box, Typography, Tabs, Tab, Card } from '@material-ui/core'
import { Skeleton } from '@material-ui/lab'
import firebase from 'firebase'
import algoliasearch from 'algoliasearch'
import UserView from '../../components/UserView'
import Post from '../../components/Post'
import InfiniteScroll from 'react-infinite-scroll-component'
import index from '../../config/algoliaConfig'

class SearchView extends Component {
 constructor(props) {
   super(props)
   this.state = {
     search: this.props.match.params.id,
     isPostLoading: true,
     isAccountLoading: true,
     isLoading: true,
     posts: [],
     accounts: [],
     postLoadLength: 10,
     accountLoadLength: 10,
     tabValue: 0,

   }
 }

 async componentDidMount() {
   if (this.state.search.length >= 25) {
     this.props.history.push('/')
     return
   }
   const response = await index.search(this.state.search)
   const hits = response.hits
   let accounts = []
   hits.forEach((h) => {
     accounts.push(h.uid)
   })
   this.setState({
     ...this.state,
     accounts: accounts,
     isAccountLoading: false,
   })
   const dbRef = firebase.firestore()
   const tagRef = dbRef.collection('tags').doc(this.state.search)
   const tagResult = await tagRef.get()

   if (tagResult.exists) {
     const tagData = tagResult.data()
     await tagRef.set({
       search: tagData.search?tagData.search:0+1
     },{merge:true})
     let posts = []
     tagData.posts.forEach((id) => {
       posts.push(id)
     })
     this.setState({
       ...this.state,
       isPostLoading: false,
       posts: posts
     })
   }

   this.setState({
     isLoading: false,
   })

 }

async componentWillReceiveProps(nextProps) {
   if (nextProps.match.params.id === this.state.search) return
   this.setState({
     search: nextProps.match.params.id,
     isLoading: true,
   })
   if (this.state.search.length >= 25) {
     this.props.history.push('/')
     return
   }
   const response = await index.search(this.state.search)
   const hits = response.hits
   let accounts = []
   hits.forEach((h) => {
     accounts.push(h.uid)
   })
   this.setState({
     ...this.state,
     accounts: accounts,
     isAccountLoading: false,
   })


   const dbRef = firebase.firestore()
   const tagRef = dbRef.collection('tags').doc(this.state.search)
   const tagResult = await tagRef.get()

   if (tagResult.exists) {
     const tagData = tagResult.data()
     await tagRef.set({
       search: tagData.search?tagData.search:0+1
     },{merge:true})
     let posts = []
     tagData.posts.forEach((id) => {
       posts.push(id)
     })
     this.setState({
       ...this.state,
       isPostLoading: false,
       posts: posts
     })
   }

   this.setState({
     isLoading: false,
   })


 }

 handleTabChange = (event, value) => {
   this.setState({
     ...this.state,
     tabValue: value
   })
 }
 loadMorePosts = () => {
   this.setState({
     ...this.state,
     postLoadLength: Math.min(this.state.postLoadLength + 10, this.state.posts.length)
   })
 }

 loadMoreAccounts = () => {
   this.setState({
     ...this.state,
     accountLoadLength: Math.min(this.state.accountLoadLength + 10,
this.state.accounts.length)
   })

 }

 render() {
   if (this.state.isLoading)
     return (
       <Grid className='profile'>
         <Grid item xs={12}>
           <Skeleton type='rect' className='profilePlaceholder' />
         </Grid>
       </Grid>
     )
   return (

     <Grid className='profile'>

       <Grid item xs={12}>
         <Paper elevation={0}>
           <Grid item xs={12}>

             <Tabs
               value={this.state.tabValue}
               onChange={this.handleTabChange}
               indicatorColor="secondary"
               textColor="secondary"
               centered
             >
               <Tab label="Posts" />
               <Tab label="Users" />
             </Tabs>
           </Grid>
         </Paper>
       </Grid>
       <Grid item xs={12}>
         {
           this.state.tabValue === 0
             ?
             this.state.posts.length === 0
             ?
             <Card variant='outlined' style={{ marginTop: 20 }}>
```

```
               <Typography style={{ textAlign: 'center' }} variant='h4'> No Matching
Posts</Typography>
             </Card>
             :
             <InfiniteScroll
               dataLength={this.state.posts.length}
               next={this.loadMorePosts}
               hasMore={true}
             >
               {this.state.posts.slice(0, this.state.postLoadLength).map((id) => {
                 return (
                   <Grid item xs={12}>
                     <Post key={id} deletePost={this.deletePost} id={id} />
                   </Grid>
                 )
               })}
             </InfiniteScroll>
             :
             this.state.accounts.length === 0
               ?
               <Card variant='outlined' style={{ marginTop: 20 }}>
                 <Typography style={{ textAlign: 'center' }} variant='h4'> No Matching
Users</Typography>
               </Card>
               :
               <InfiniteScroll
                 dataLength={this.state.posts.length}
                 next={this.loadMorePosts}
                 hasMore={true}
               >
                 {this.state.accounts.slice(0,
this.state.accountLoadLength).map((account) => {
                   return (
                     <Grid item xs={12}>
                       <UserView id={account} />
                     </Grid>
                   )
                 })}
               </InfiniteScroll>


         }
       </Grid>

     </Grid>
   )
 }
```

```
}

export default withRouter(SearchView);
```

**utils/compress.js**

```
// From https://developer.mozilla.org/en-US/docs/Web/API/HTMLCanvasElement/toBlob, needed
for Safari:
if (!HTMLCanvasElement.prototype.toBlob) {
   Object.defineProperty(HTMLCanvasElement.prototype, 'toBlob', {
       value: function(callback, type, quality) {

           var binStr = atob(this.toDataURL(type, quality).split(',')[1]),
               len = binStr.length,
               arr = new Uint8Array(len);

           for (var i = 0; i < len; i++) {
               arr[i] = binStr.charCodeAt(i);
           }

           callback(new Blob([arr], {type: type || 'image/png'}));
       }
   });
}

window.URL = window.URL || window.webkitURL;

// Modified from https://stackoverflow.com/a/32490603, cc by-sa 3.0
// -2 = not jpeg, -1 = no data, 1..8 = orientations
function getExifOrientation(file, callback) {
   // Suggestion from
http://code.flickr.net/2012/06/01/parsing-exif-client-side-using-javascript-2/:
   if (file.slice) {
       file = file.slice(0, 131072);
   } else if (file.webkitSlice) {
       file = file.webkitSlice(0, 131072);
   }

   var reader = new FileReader();
   reader.onload = function(e) {
       var view = new DataView(e.target.result);
       if (view.getUint16(0, false) != 0xFFD8) {
           callback(-2);
           return;
       }
       var length = view.byteLength, offset = 2;
       while (offset < length) {
           var marker = view.getUint16(offset, false);
           offset += 2;
           if (marker == 0xFFE1) {
               if (view.getUint32(offset += 2, false) != 0x45786966) {
                   callback(-1);
                   return;
               }
               var little = view.getUint16(offset += 6, false) == 0x4949;
               offset += view.getUint32(offset + 4, little);
               var tags = view.getUint16(offset, little);
               offset += 2;
               for (var i = 0; i < tags; i++)
                   if (view.getUint16(offset + (i * 12), little) == 0x0112) {
                       callback(view.getUint16(offset + (i * 12) + 8, little));
                       return;
                   }
           }
           else if ((marker & 0xFF00) != 0xFF00) break;
           else offset += view.getUint16(offset, false);
       }
       callback(-1);
   };
   reader.readAsArrayBuffer(file);
}

// Derived from https://stackoverflow.com/a/40867559, cc by-sa
function imgToCanvasWithOrientation(img, rawWidth, rawHeight, orientation) {
   var canvas = document.createElement('canvas');
   if (orientation > 4) {
       canvas.width = rawHeight;
       canvas.height = rawWidth;
   } else {
       canvas.width = rawWidth;
       canvas.height = rawHeight;
   }

   if (orientation > 1) {
       console.log("EXIF orientation = " + orientation + ", rotating picture");
   }

   var ctx = canvas.getContext('2d');
   switch (orientation) {
       case 2: ctx.transform(-1, 0, 0, 1, rawWidth, 0); break;
       case 3: ctx.transform(-1, 0, 0, -1, rawWidth, rawHeight); break;
       case 4: ctx.transform(1, 0, 0, -1, 0, rawHeight); break;
       case 5: ctx.transform(0, 1, 1, 0, 0, 0); break;
       case 6: ctx.transform(0, 1, -1, 0, rawHeight, 0); break;
       case 7: ctx.transform(0, -1, -1, 0, rawHeight, rawWidth); break;
       case 8: ctx.transform(0, -1, 1, 0, 0, rawWidth); break;
   }
   ctx.drawImage(img, 0, 0, rawWidth, rawHeight);
   return canvas;
}

function reduceFileSize(file, acceptFileSize, maxWidth, maxHeight, quality, callback) {
   if (file.size <= acceptFileSize) {
       callback(file);
       return;
   }
   var img = new Image();
   img.onerror = function() {
       URL.revokeObjectURL(this.src);
       callback(file);
   };
   img.onload = function() {
       URL.revokeObjectURL(this.src);
       getExifOrientation(file, function(orientation) {
           var w = img.width, h = img.height;
           var scale = (orientation > 4 ?
               Math.min(maxHeight / w, maxWidth / h, 1) :
               Math.min(maxWidth / w, maxHeight / h, 1));
           h = Math.round(h * scale);
           w = Math.round(w * scale);

           var canvas = imgToCanvasWithOrientation(img, w, h, orientation);
           canvas.toBlob(function(blob) {
               callback(blob);
           }, 'image/jpeg', quality);
       });
   };
   img.src = URL.createObjectURL(file);
}

export default reduceFileSize
```

**App.js:**

```
import './App.css';
import React, { useState } from 'react';
import { BrowserRouter, Switch, Route, NotFoundRoute } from 'react-router-dom';
import useMediaQuery from '@material-ui/core/useMediaQuery';
import { createMuiTheme, ThemeProvider } from '@material-ui/core/styles';
import { CssBaseline } from '@material-ui/core';
import Navbar from './components/Navbar';
import Footer from './components/Footer'
import Home from './routes/Home';
import Profile from './routes/Profile';
import OurStory from './routes/OurStory';
import SearchView from './routes/SearchView';
import './App.scss';
import { StylesProvider } from '@material-ui/core/styles';



function App() {
 const lightTheme = createMuiTheme({
   palette: {
     type: 'light'
   },
 })
 const darkTheme = createMuiTheme({
   palette: {
     type: 'dark'
   },
 })


 const [themeState, setTheme] = useState(lightTheme);
 const [themeString, setThemeString] = useState('light');

 const toggleDarkMode = () => {
   if (themeString === 'light') {
     setThemeString('dark')
     setTheme(darkTheme)
   }else{
     setThemeString('light')
     setTheme(lightTheme)

   }
 }

 return (
   <BrowserRouter>
     <ThemeProvider theme={themeState}>
       <CssBaseline />
       <StylesProvider injectFirst>
         <Navbar toggleDarkMode={toggleDarkMode} />
         <Switch>
           <Route exact path='/' component={Home} />
           <Route exact path='/search/:id' render={(props) => <SearchView {...props} />}
/>

           <Route path='/profile/:id' render={(props) => <Profile {...props} />} />
           <Route path='/ourstory' component={OurStory} />
           <Route component={Home} />
         </Switch>
         <Footer />
       </StylesProvider>
     </ThemeProvider>
   </BrowserRouter>
 );
}

export default App;
```

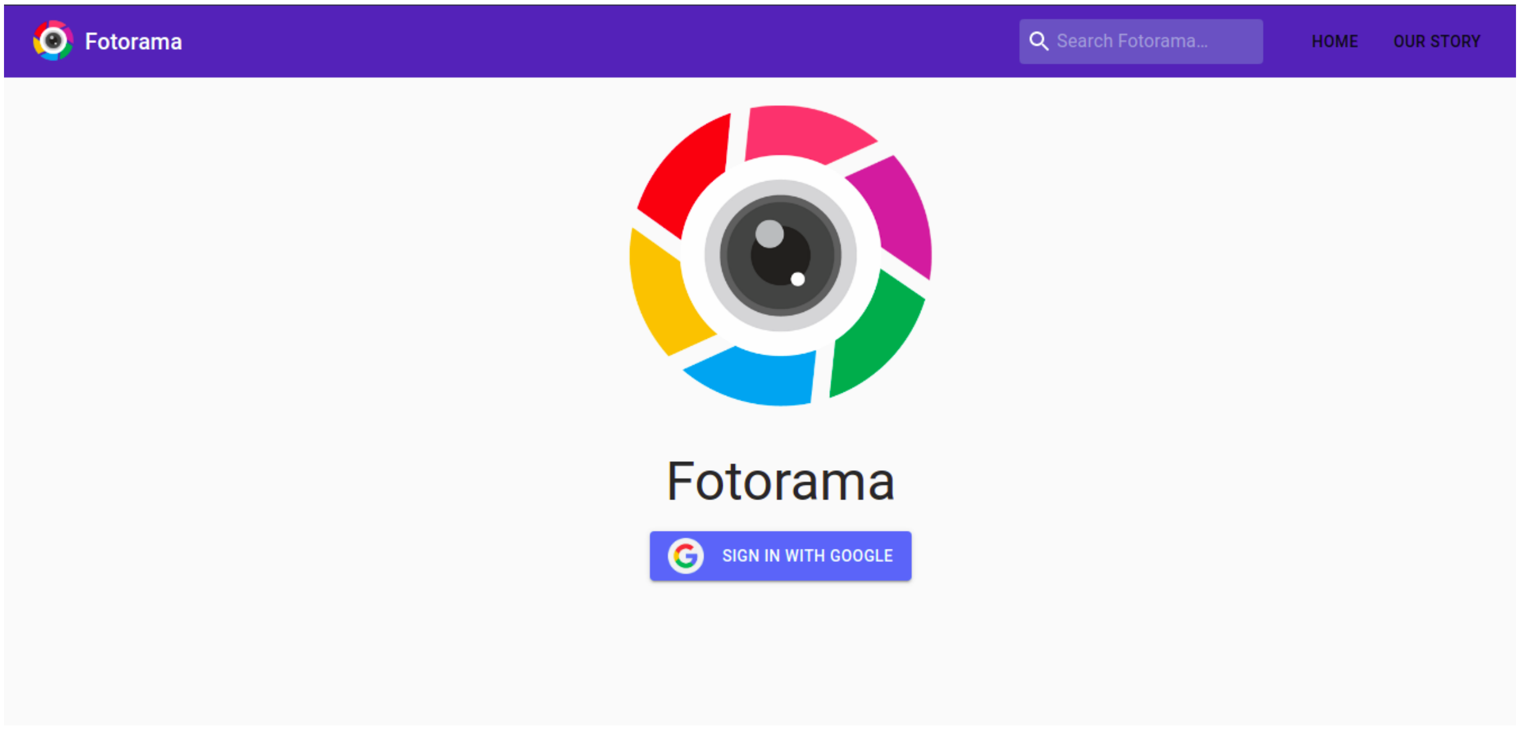
**Index.js:**

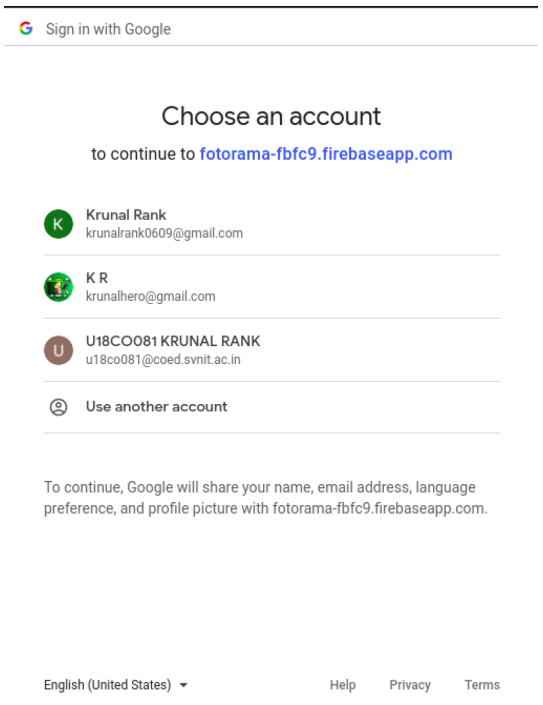
```
import React from 'react'
import ReactDOM from 'react-dom'
import './index.css'
import App from './App'
import registerServiceWorker from './registerServiceWorker'
import firebase from 'firebase'
import firebaseConfig from './config/firebaseConfig'

firebase.initializeApp(firebaseConfig)
firebase.auth().setPersistence(firebase.auth.Auth.Persistence.SESSION);
firebase.firestore().enablePersistence()

ReactDOM.render(
 <App />,
 document.getElementById('root')
);
registerServiceWorker();
```

## Screenshots:




Sign in with Google

Choose an account

to continue to fotorama-fbfc9.firebaseapp.com

Krunal Rank
krunalrank0609@gmail.com

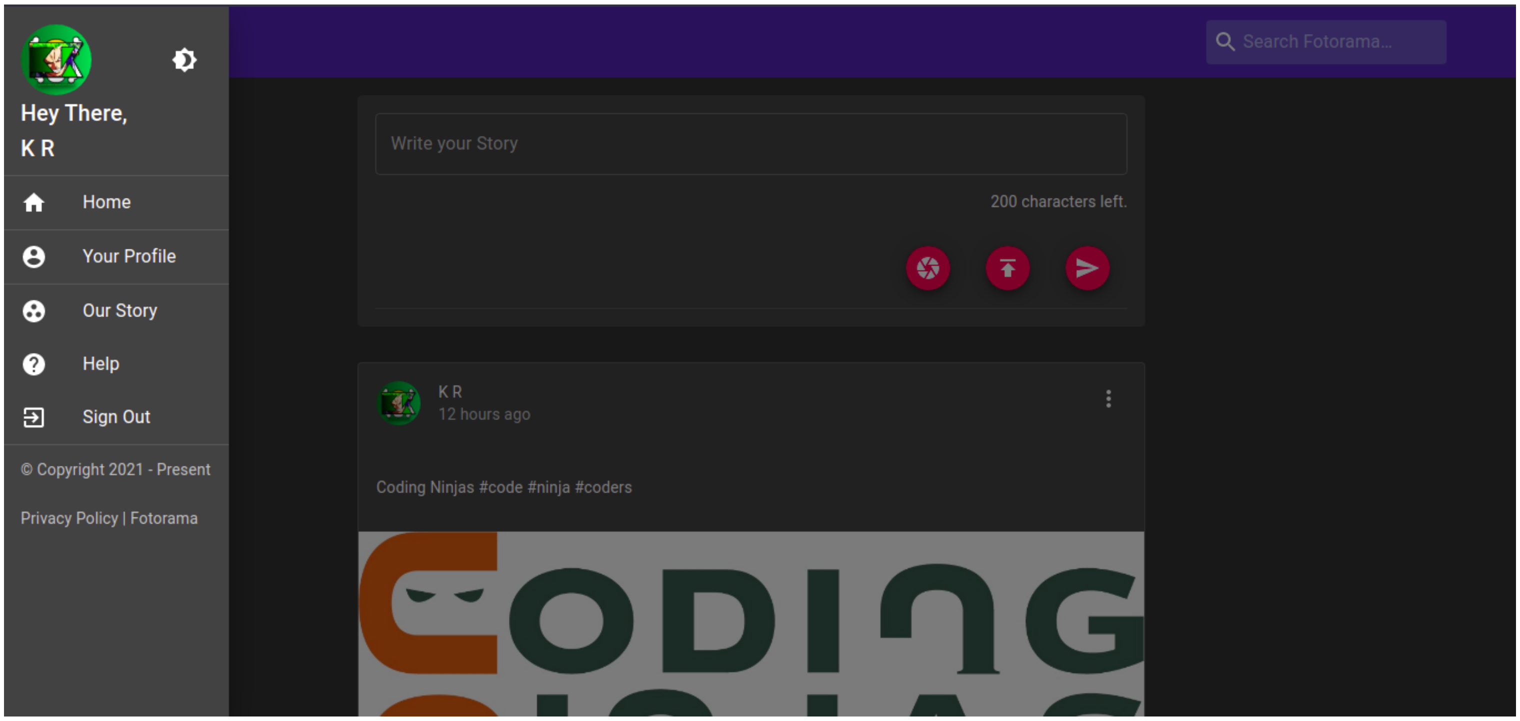
K R
krunalhero@gmail.com

U18CO081 KRUNAL RANK
u18co081@coed.svnit.ac.in

Use another account

To continue, Google will share your name, email address, language preference, and profile picture with fotorama-fbfc9.firebaseapp.com.

English (United States) Help Privacy Terms


Hey There,
K R

© Copyright 2021 - Present

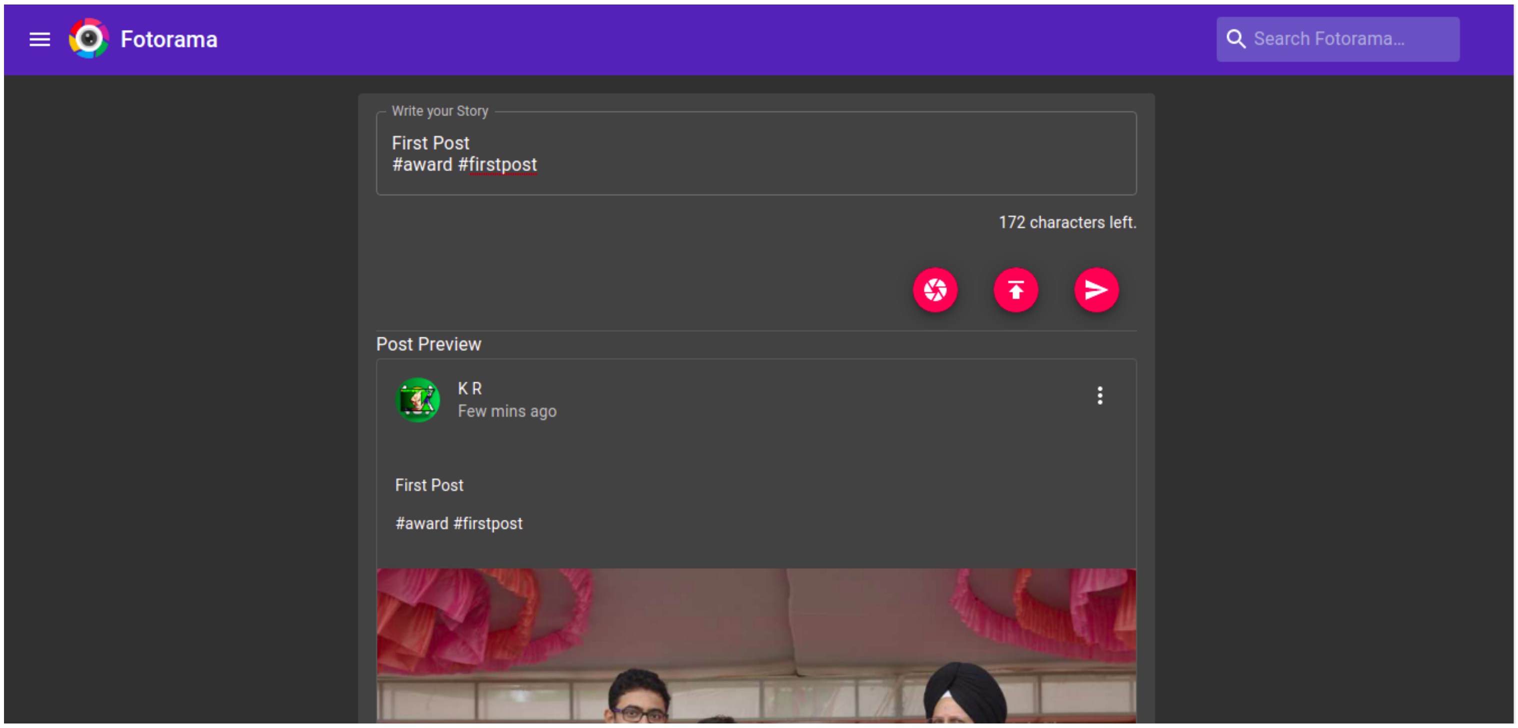
Privacy Policy | Fotorama

Write your Story

200 characters left.

K R
12 hours ago

Coding Ninjas #code #ninja #coders



Write your Story
First Post
#award #firstpost

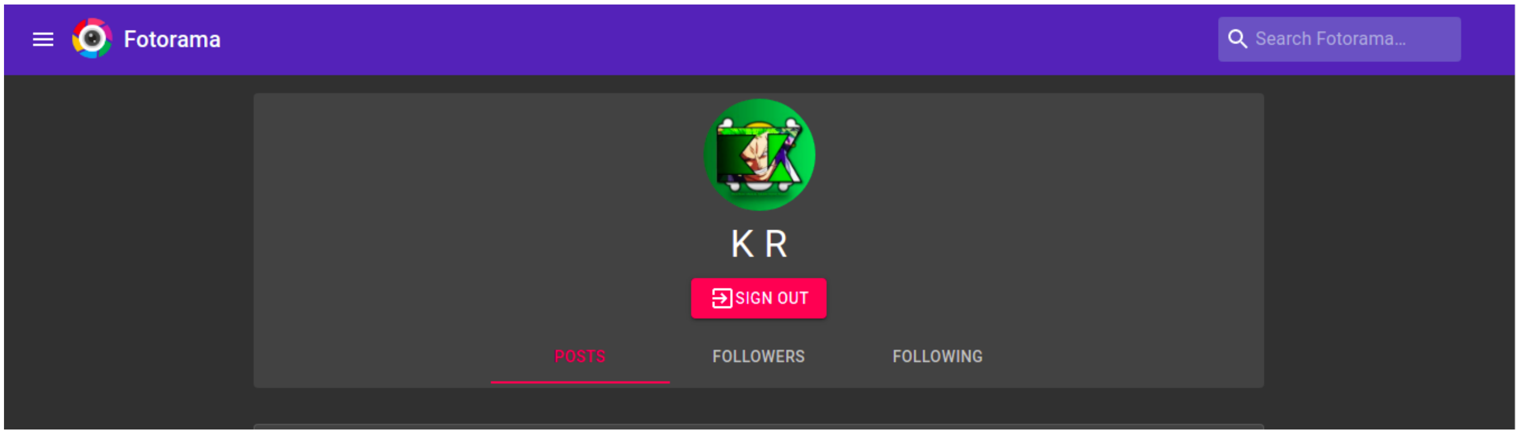
172 characters left.

Post Preview

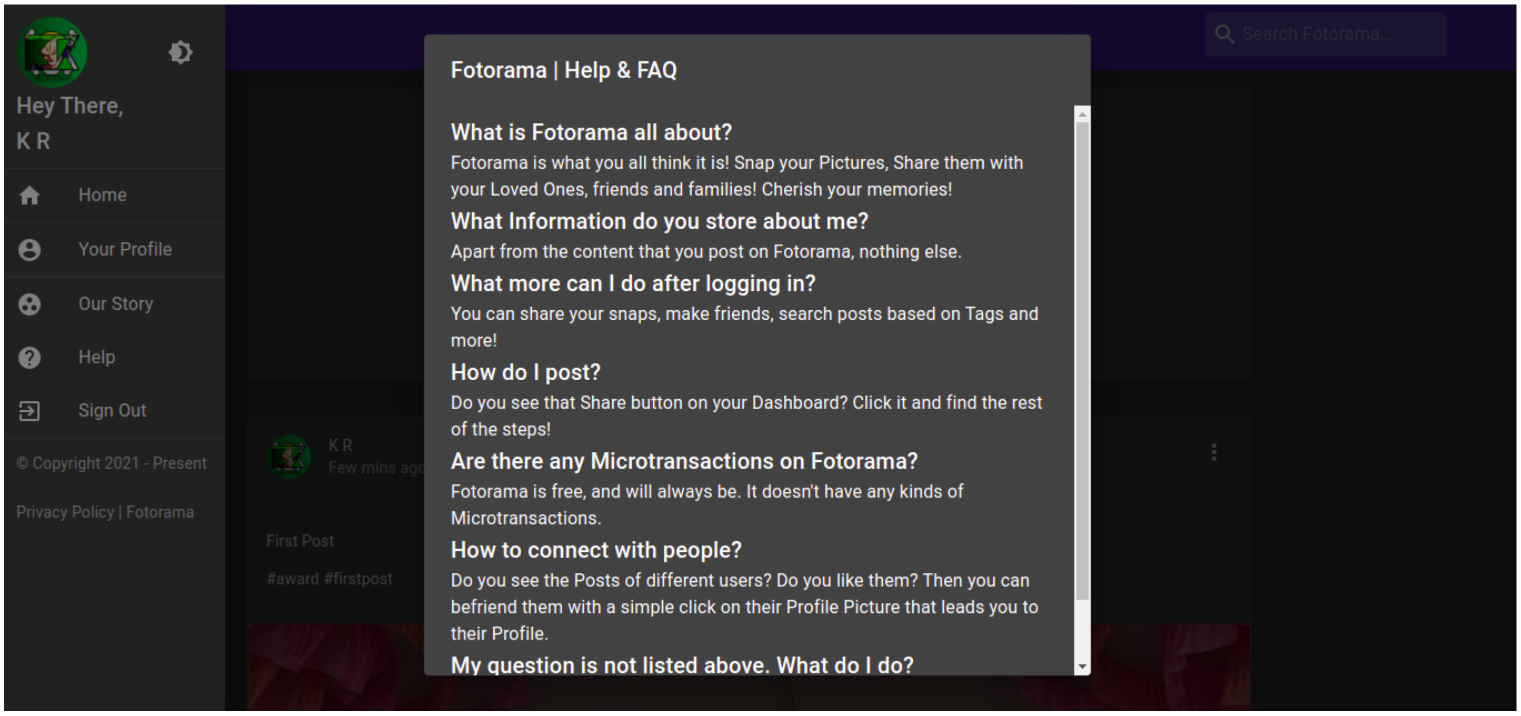
K R
Few mins ago

First Post

#award #firstpost



K R



Fotorama | Help & FAQ

**What is Fotorama all about?**
Fotorama is what you all think it is! Snap your Pictures, Share them with your Loved Ones, friends and families! Cherish your memories!

**What Information do you store about me?**
Apart from the content that you post on Fotorama, nothing else.

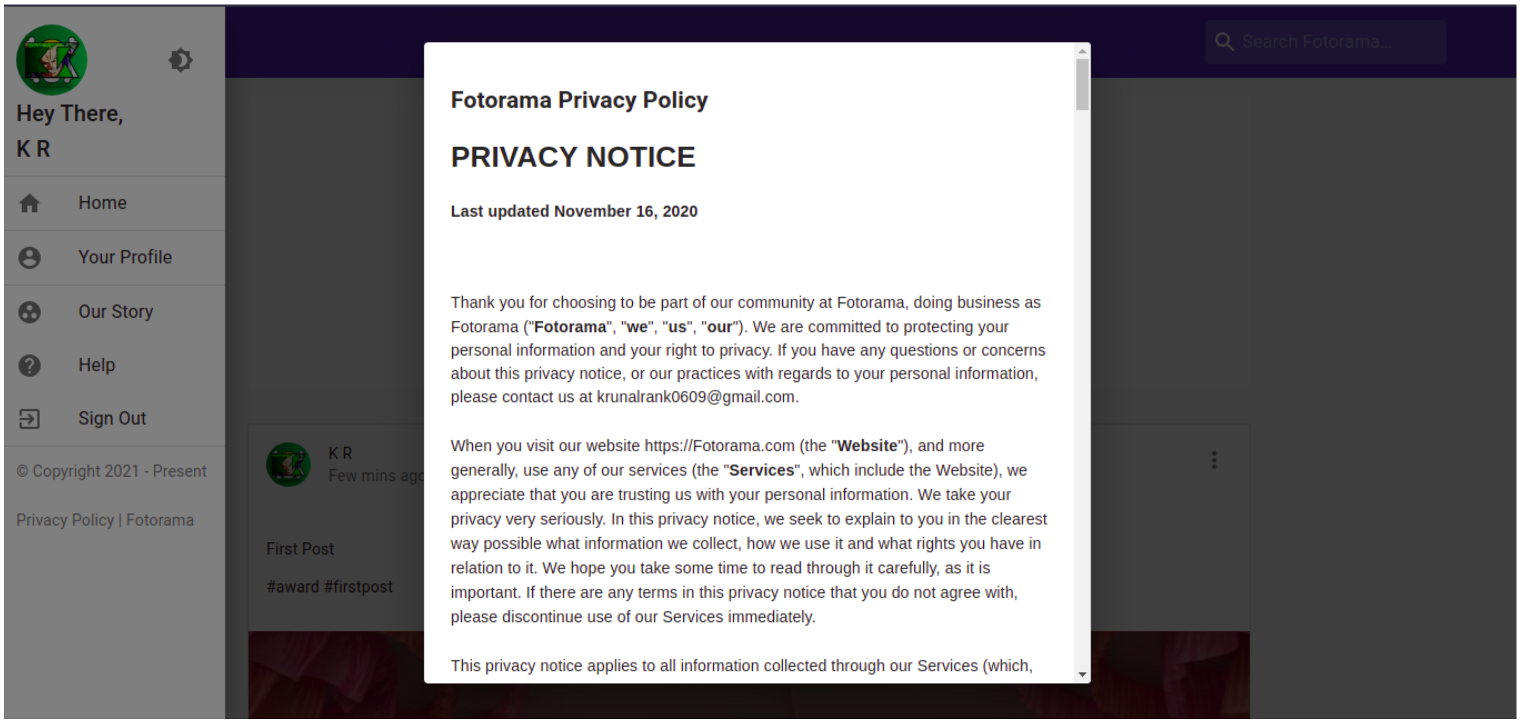
**What more can I do after logging in?**
You can share your snaps, make friends, search posts based on Tags and more!

**How do I post?**
Do you see that Share button on your Dashboard? Click it and find the rest of the steps!

**Are there any Microtransactions on Fotorama?**
Fotorama is free, and will always be. It doesn't have any kinds of Microtransactions.

**How to connect with people?**
Do you see the Posts of different users? Do you like them? Then you can befriend them with a simple click on their Profile Picture that leads you to their Profile.

**My question is not listed above. What do I do?**



**Fotorama Privacy Policy**

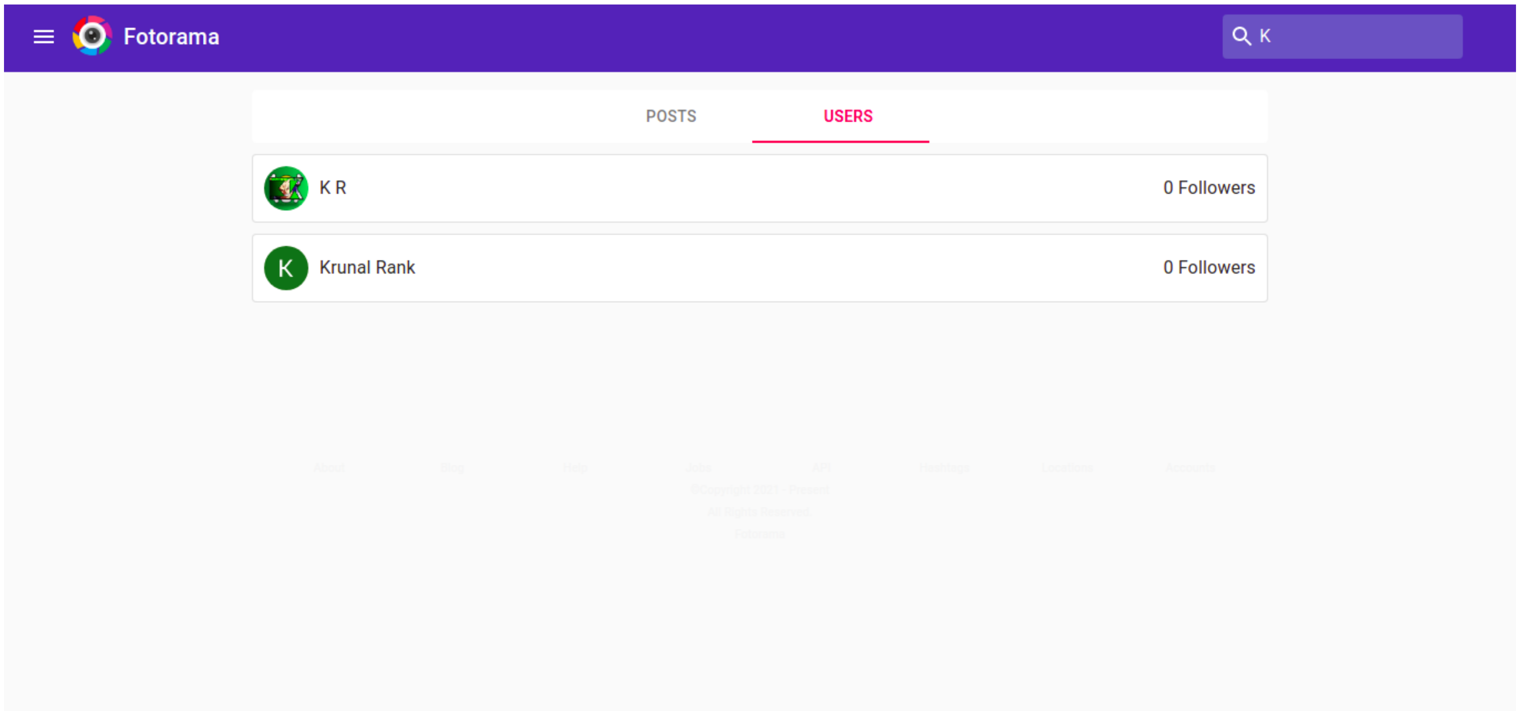
**PRIVACY NOTICE**

**Last updated November 16, 2020**

Thank you for choosing to be part of our community at Fotorama, doing business as Fotorama ("**Fotorama**", "**we**", "**us**", "**our**"). We are committed to protecting your personal information and your right to privacy. If you have any questions or concerns about this privacy notice, or our practices with regards to your personal information, please contact us at krunalrank0609@gmail.com.

When you visit our website https://Fotorama.com (the "**Website**"), and more generally, use any of our services (the "**Services**", which include the Website), we appreciate that you are trusting us with your personal information. We take your privacy very seriously. In this privacy notice, we seek to explain to you in the clearest way possible what information we collect, how we use it and what rights you have in relation to it. We hope you take some time to read through it carefully, as it is important. If there are any terms in this privacy notice that you do not agree with, please discontinue use of our Services immediately.

This privacy notice applies to all information collected through our Services (which,



K R — 0 Followers

Krunal Rank — 0 Followers