Task 1:

Create a new resource named "news" in mockapi.io by signing up and creating a new project. Then, create a new collection named "news" and define its schema.

Edit/replace data for the news resource by copying the content of data.json file and pasting it into the collection in mockapi.io.

Task 2:

3. Install the react-router-dom library and create a Navbar component with links to the Home, Top News, Contact, and Sign In routes.

At the Home route, fetch the news data from the news resource in mockapi.io using the useEffect hook and display only the news with the "attractive" property set to true. Use the Link component to switch to the Detail route when a user clicks on the title link or detail button.

At the Detail route, fetch the details of the news from the news resource in mockapi.io using the useParams hook and display all the information about the news.

Implement the Google login with Firebase authentication. After successful login, store the user information in the app's state and redirect to the Dashboard route. Update the Sign In link in the Navbar to show the user's email.

At the Dashboard route, fetch the news data from the news resource in mockapi.io and display it in a table. Each row should have links or icons for CRUD actions.

At the Add route, create a form with validation for all fields, including description and content as text areas, created as a date, views as a number with a default value of 1, and attractive as a checkbox. On submission, send a POST request to add the news to the news resource in mockapi.io.

At the Dashboard route, implement the delete action by displaying a confirmation modal before deleting the news. After successful deletion, display a notice and render the new list of news.

At the Dashboard route, implement the update action by displaying a form with pre-filled data for the selected news. On submission, send a PUT request to update the news in the news resource in mockapi.io. Include validation for all fields.

TASK 1/For this task, we will create a React application, set up a news resource on mockapi.io and fetch data from it to display on our application. We will also create a navbar and two routes: Home and Detail. On the Home page, we will display all the news articles that have "attractive" set to true. When a user clicks on a news article, they will be taken to the Detail page, where they can view all the details of the article.

Step 1: Create a new React application using create-react-app

Open your terminal and run the following command:

npx create-react-app your-rollnumber

This will create a new React application in a folder named "your-rollnumber".

Step 2: Set up a news resource on mockapi.io and declare a Schema

Go to mockapi.io and sign up for a free account. Once you are signed in, create a new resource named "news" and declare the following schema:

```
{
  "title": "string",
  "description": "string",
  "content": "string",
  "img": "string",
  "created": "string",
  "views": "number",
  "attractive": "boolean"
}
```

Step 3: Edit/replace data for news resource

Copy the content of the data.json file and paste it into the "news" resource on mockapi.io.

Step 4: Create a Navbar for navigating all the routes in your application

In your React application, install the react-router-dom library by running the following command in your terminal:

npm install react-router-dom

Next, open the src/App.js file and import the BrowserRouter, Switch, and Route components from react-router-dom:

```
import { BrowserRouter, Switch, Route } from 'react-router-dom';
```

In the render method of the App component, wrap the contents of the div with the BrowserRouter component:

```
render() {
  return (
    <BrowserRouter>
      <div className="App">
        ...
      </div>
    </BrowserRouter>
  );
}
```

Create a new component called Navbar and add it to the render method of the App component:

```
import { Link } from 'react-router-dom';
```

```
function Navbar() {
  return (
    <nav>
      <ul>
        <li>
          <Link to="/">Home</Link>
        </li>
        <li>
          <Link to="/top-news">Top News</Link>
        </li>
        <li>
          <Link to="/contact">Contact</Link>
        </li>
        <li>
          <Link to="/sign-in">Sign in</Link>
        </li>
      </ul>
    </nav>
  );
}

render() {
  return (
    <BrowserRouter>
      <div className="App">
        <Navbar />
        <Switch>
          <Route exact path="/" component={Home} />
          <Route path="/detail/:id" component={Detail} />
```

```
      </Switch>
    </div>
  </BrowserRouter>
 );
}
```

STEP 5/To display all the information of news on the detail route, we can create a new component named NewsDetail and use it in the detail route. Here's an example implementation:

Javascript

```
import { useState, useEffect } from 'react';
import { useParams } from 'react-router-dom';
import axios from 'axios';

function NewsDetail() {
 const { id } = useParams();
 const [news, setNews] = useState(null);

 useEffect(() => {
  axios.get(`https://mockapi.io/api/v1/news/${id}`)
   .then(response => setNews(response.data))
   .catch(error => console.log(error));
 }, [id]);

 if (!news) {
  return <div>Loading...</div>;
 }
```

```
  return (
    <div>
      <h1>{news.title}</h1>
      <img src={news.img} alt={news.title} />
      <p>{news.description}</p>
      <p>{news.content}</p>
      <p>Created: {news.created}</p>
      <p>Views: {news.views}</p>
      <p>Attractive: {news.attractive ? 'Yes' : 'No'}</p>
    </div>
  );
}

export default NewsDetail;
```

In this implementation, we first import the necessary modules - useState, useEffect, useParams, and axios. useParams allows us to retrieve the id parameter from the URL. We then use useState to keep track of the news data and set its initial value to null. We use useEffect to fetch the news data from the API based on the id parameter in the URL. If the news data is not yet available, we display a simple loading message. Once the news data is available, we render its properties in the JSX, including its title, img, description, content, created, views, and attractive properties.

To use this component in the detail route, we can import it and add a new route to our App.js file:

```
import { BrowserRouter as Router, Switch, Route, Link } from 'react-router-dom';
import Home from './components/Home';
import TopNews from './components/TopNews';
import Contact from './components/Contact';
import SignIn from './components/SignIn';
import AddNews from './components/AddNews';
import Dashboard from './components/Dashboard';
```

```
import NewsDetail from './components/NewsDetail';

function App() {
  return (
    <Router>
      <div>
        <nav>
          <ul>
            <li>
              <Link to="/">Home</Link>
            </li>
            <li>
              <Link to="/top-news">Top News</Link>
            </li>
            <li>
              <Link to="/contact">Contact</Link>
            </li>
            <li>
              <Link to="/sign-in">Sign in</Link>
            </li>
          </ul>
        </nav>

        <Switch>
          <Route path="/top-news">
            <TopNews />
          </Route>
          <Route path="/contact">
            <Contact />
```

```
        </Route>
        <Route path="/sign-in">
          <SignIn />
        </Route>
        <Route path="/add-news">
          <AddNews />
        </Route>
        <Route path="/dashboard">
          <Dashboard />
        </Route>
        <Route path="/news/:id">
          <NewsDetail />
        </Route>
        <Route path="/">
          <Home />
        </Route>
      </Switch>
    </div>
  </Router>
 );
}

export default App;
import React, { useState, useEffect } from 'react';
import { useParams } from 'react-router-dom';
import axios from 'axios';

const Detail = () => {
  const [news, setNews] = useState({});
```

```
  const { id } = useParams();

  useEffect(() => {
    axios.get(`https://613a6ea40f1e000017a454d8.mockapi.io/api/v1/news/${id}`)
      .then(response => {
        setNews(response.data);
      })
      .catch(error => {
        console.log(error);
      });
  }, [id]);

  return (
    <div className="container">
      <h1>{news.title}</h1>
      <img src={news.img} alt={news.title} />
      <p>{news.description}</p>
      <p>{news.content}</p>
      <p>Created: {news.created}</p>
      <p>Views: {news.views}</p>
      <p>Attractive: {news.attractive ? 'Yes' : 'No'}</p>
    </div>
  );
};

export default Detail;
```

In this implementation, we use the useParams hook from react-router-dom to extract the id parameter from the URL. We then make an API call to retrieve the news item with that id using Axios. Once we have the news item, we render the relevant information on the page.

Note that this implementation assumes that the views count is stored as a number in the API response. If it is stored as a string, you may need to convert it to a number before rendering it on the page.

Task 2. Implement some actions in private routes.

6/To build the login with Google account action, we need to use Google Sign-In API which allows users to sign in to the application using their Google account.

First, we need to create a Google Cloud Platform project and enable the Google Sign-In API. Then, we can obtain a client ID which is needed to use the API.

Next, we need to install the Google Sign-In library for the frontend. We can use the following command to install it:

npm install react-google-login

After installing the library, we can use it to add a Google Sign-In button to the application. When the user clicks on the button, they will be prompted to sign in with their Google account. Once they have successfully signed in, we can obtain their user profile information and use it to authenticate them in our application.

Here's an example of how to use the Google Sign-In library in a React component:

```
import React from 'react';
import { GoogleLogin } from 'react-google-login';
```

```
const CLIENT_ID = 'your-client-id';

const Login = () => {
  const onSuccess = (response) => {
    console.log(response.profileObj);
    // authenticate the user in your application
    // redirect to dashboard route
  };

  const onFailure = (response) => {
    console.log(response);
  };

  return (
    <GoogleLogin
      clientId={CLIENT_ID}
      onSuccess={onSuccess}
      onFailure={onFailure}
      cookiePolicy={'single_host_origin'}
      buttonText="Sign in with Google"
    />
  );
};

export default Login;
```

In the onSuccess callback function, we can obtain the user profile information from the response object and use it to authenticate the user in our application. We can then redirect them to the dashboard route using react-router-dom library's useHistory hook.

To replace the Sign in link of Navbar with the Google Sign-In button, we can simply replace the a element with the Login component we created above.

7/To display the list of news in the dashboard route and provide CRUD actions, we can create a new component NewsTable that takes a list of news as input and renders a table with each news item as a row. For each row, we can provide links or icons for CRUD actions like edit and delete. Here's an example implementation:

```
import React from 'react';

import { Link } from 'react-router-dom';


const NewsTable = ({ newsList, onDelete }) => {

 return (

  <table>

   <thead>

    <tr>

     <th>Title</th>

     <th>Description</th>

     <th>Created</th>

     <th>Views</th>

     <th>Attractive</th>

     <th>Actions</th>

    </tr>

   </thead>

   <tbody>

    {newsList.map((news) => (

     <tr key={news.id}>

      <td>{news.title}</td>

      <td>{news.description}</td>

      <td>{news.created}</td>
```

```
          <td>{news.views}</td>
          <td>{news.attractive ? 'Yes' : 'No'}</td>
          <td>
            <Link to={`/edit/${news.id}`}>Edit</Link>
            <button onClick={() => onDelete(news.id)}>Delete</button>
          </td>
        </tr>
      ))}
    </tbody>
  </table>
 );
};

export default NewsTable;
```

In the above component, we are rendering a table with the columns Title, Description, Created, Views, Attractive and Actions. For each news item in the newsList, we are creating a table row with columns displaying the relevant information and providing links or buttons for CRUD actions. The Link component is used to link to the edit route for a specific news item, while the onDelete function is called when the delete button is clicked for a news item.

To use this component in the dashboard route, we can import it and render it like this:

```
import React, { useState, useEffect } from 'react';
import { auth } from '../firebase';
import { useHistory } from 'react-router-dom';
import NewsTable from '../components/NewsTable';
import { getNews, deleteNews } from '../api/news';

const Dashboard = () => {
```

```
const [newsList, setNewsList] = useState([]);
const history = useHistory();

useEffect(() => {
  const unsubscribe = auth.onAuthStateChanged((user) => {
    if (!user) {
      history.push('/signin');
    }
  });

  return () => {
    unsubscribe();
  };
}, [history]);

useEffect(() => {
  const fetchData = async () => {
    const newsData = await getNews();
    setNewsList(newsData);
  };

  fetchData();
}, []);

const handleDelete = async (id) => {
  await deleteNews(id);
  setNewsList((prevNews) => prevNews.filter((news) => news.id !== id));
  // show notice here
};
```

```
  return (
    <div>
      <h1>Dashboard</h1>
      <NewsTable newsList={newsList} onDelete={handleDelete} />
    </div>
  );
};

export default Dashboard;
```

In the above code, we are first checking if the user is logged in, and if not, redirecting to the sign in route. We are also fetching the list of news using the getNews function from the API and setting the state of the newsList variable using the setNewsList function.

We are passing the newsList and handleDelete function

8/To create the Add route with the requirements, we can use a form with different types of input fields for each data attribute of the news. We will also need to use some libraries for validation and for displaying the date picker.

Here is an example code snippet for the Add route:

```
import React, { useState } from 'react';
import { useHistory } from 'react-router-dom';
import { useForm } from 'react-hook-form';
import DatePicker from 'react-datepicker';
import 'react-datepicker/dist/react-datepicker.css';

const AddNews = ({ userToken }) => {
```

```
const history = useHistory();
const { register, handleSubmit, formState: { errors } } = useForm();
const [createdDate, setCreatedDate] = useState(new Date());
const [status, setStatus] = useState(false);
const [attractive, setAttractive] = useState(false);

const onSubmit = async (data) => {
 // send data to server
 const response = await fetch('https://61461cb71d16670017b3f86a.mockapi.io/api/v1/news', {
  method: 'POST',
  headers: {
   'Content-Type': 'application/json',
   'Authorization': `Bearer ${userToken}`
  },
  body: JSON.stringify({
   title: data.title,
   description: data.description,
   content: data.content,
   img: data.img,
   created: createdDate,
   views: parseInt(data.views),
   status: status,
   attractive: attractive
  })
 });

 if (response.ok) {
  // show success message and redirect to dashboard
  alert('News added successfully!');
```

```
    history.push('/dashboard');
   } else {
    // show error message
    alert('An error occurred while adding the news. Please try again later.');
   }
  };

  return (
   <div className="container">
    <h1>Add News</h1>
    <form onSubmit={handleSubmit(onSubmit)}>
     <div className="form-group">
      <label htmlFor="title">Title</label>
      <input type="text" className="form-control" id="title" {...register("title", { required: true })} />
      {errors.title && <span className="text-danger">This field is required</span>}
     </div>
     <div className="form-group">
      <label htmlFor="description">Description</label>
      <textarea className="form-control" id="description" {...register("description", { required:
true })}></textarea>
      {errors.description && <span className="text-danger">This field is required</span>}
     </div>
     <div className="form-group">
      <label htmlFor="content">Content</label>
      <textarea className="form-control" id="content" {...register("content", { required:
true })}></textarea>
      {errors.content && <span className="text-danger">This field is required</span>}
     </div>
     <div className="form-group">
```

```
  <label htmlFor="img">Image URL</label>
  <input type="text" className="form-control" id="img" {...register("img", { required: true })} />
  {errors.img && <span className="text-danger">This field is required</span>}
 </div>
 <div className="form-group">
  <label htmlFor="created">Created</label>
  <br />
  <DatePicker selected={createdDate} onChange={date => setCreatedDate(date)} />
 </div>
 <div className="form-group">
  <label htmlFor="views">Views</label>
  <input type="number" className="form-control" id="views" {...register("views", { required: true }
```

To fulfill the requirements of the Add route, you can follow these steps:

Create a new route named "Add" and set its path to "/add".

Inside the Add component, create a form using React's form elements such as input, textarea, and select to collect data from the user.

Use state hooks to store the form data and validation errors.

Implement form validation using conditional rendering of error messages and/or input styling.

Use the Axios library to send a POST request to the news endpoint with the form data when the user submits the form.

Display a success message and clear the form after a successful submission.

Add a cancel button that redirects the user to the dashboard route.

Here's some sample code for the Add component:

```
import { useState } from "react";
import axios from "axios";
```

```
const Add = () => {
  const [formData, setFormData] = useState({
    title: "",
    description: "",
    content: "",
    img: "",
    created: "",
    views: 1,
    attractive: false,
  });

  const [errors, setErrors] = useState({});

  const handleChange = (e) => {
    setFormData({
      ...formData,
      [e.target.name]: e.target.value,
    });
  };

  const handleCheckboxChange = (e) => {
    setFormData({
      ...formData,
      [e.target.name]: e.target.checked,
    });
  };

  const handleSubmit = (e) => {
    e.preventDefault();
```

```
  axios
    .post("https://61339ab2d1b6700017efb5e6.mockapi.io/api/v1/news", formData)
    .then((response) => {
      alert("News added successfully!");
      setFormData({
        title: "",
        description: "",
        content: "",
        img: "",
        created: "",
        views: 1,
        attractive: false,
      });
      setErrors({});
    })
    .catch((error) => {
      setErrors(error.response.data);
    });
};

return (
  <div>
    <h2>Add News</h2>
    <form onSubmit={handleSubmit}>
      <div>
        <label>Title</label>
        <input
          type="text"
          name="title"
```

```
    value={formData.title}
    onChange={handleChange}
   />
   {errors.title && <div className="error">{errors.title}</div>}
  </div>
  <div>
   <label>Description</label>
   <textarea
    name="description"
    value={formData.description}
    onChange={handleChange}
   ></textarea>
   {errors.description && (
    <div className="error">{errors.description}</div>
   )}
  </div>
  <div>
   <label>Content</label>
   <textarea
    name="content"
    value={formData.content}
    onChange={handleChange}
   ></textarea>
   {errors.content && <div className="error">{errors.content}</div>}
  </div>
  <div>
   <label>Image URL</label>
   <input
    type="text"
```

```
          name="img"
          value={formData.img}
          onChange={handleChange}
        />
        {errors.img && <div className="error">{errors.img}</div>}
      </div>
      <div>
        <label>Created Date</label>
        <input
          type="date"
          name="created"
          value={formData.created}
          onChange={handleChange}
        />
        {errors.created && <div className="error">{errors.created}</div>}
      </div>
      <div>
        <label>Views</label>
        <input
```

9/To implement the delete action, we first need to create a function to handle the deletion of a news item. We can do this by making a DELETE request to the API using the news item's ID.

Here is an example of how we can implement the delete function in our Dashboard component:

```
import { useState, useEffect } from "react";
import axios from "axios";

function Dashboard() {
```

```
const [news, setNews] = useState([]);

useEffect(() => {
  // Fetch the news data from the API on component mount
  axios
    .get("https://mockapi.io/api/v1/news")
    .then((response) => setNews(response.data))
    .catch((error) => console.log(error));
}, []);

const handleDelete = (id) => {
  if (window.confirm("Are you sure you want to delete this news item?")) {
    axios
      .delete(`https://mockapi.io/api/v1/news/${id}`)
      .then((response) => {
        // If the delete request is successful, remove the news item from the state
        const updatedNews = news.filter((item) => item.id !== id);
        setNews(updatedNews);
        alert("News item deleted successfully.");
      })
      .catch((error) => console.log(error));
  }
};

return (
  <div>
    <h1>Dashboard</h1>
    <table>
      <thead>
```

```
        <tr>
          <th>Title</th>
          <th>Description</th>
          <th>Actions</th>
        </tr>
      </thead>
      <tbody>
        {news.map((item) => (
          <tr key={item.id}>
            <td>{item.title}</td>
            <td>{item.description}</td>
            <td>
              <button onClick={() => handleDelete(item.id)}>Delete</button>
            </td>
          </tr>
        ))}
      </tbody>
    </table>
  </div>
 );
}

export default Dashboard;
```

In this code, we first fetch the news data from the API and store it in the state using the useState and useEffect hooks. We then define the handleDelete function, which takes an ID as its parameter and confirms whether the user wants to delete the news item. If the user confirms, we make a DELETE request to the API using the news item's ID and remove the item from the state if the request is successful.

Finally, we render a table with the news items and a delete button for each item. Clicking the delete button calls the handleDelete function with the news item's ID.

10/To implement the update action, we need to create an Edit route that displays a form with the existing values of the news item to be updated. Once the form is submitted, we can make a PUT request to the API to update the news item.

Here are the steps to implement the update action:

Create a new component named EditNews that will be used to display the form for updating a news item. This component will be similar to the AddNews component but with some modifications to pre-populate the form fields with the existing values of the news item to be updated.

```
import { useState, useEffect } from 'react';
import { useHistory, useParams } from 'react-router-dom';
import axios from 'axios';

const EditNews = () => {
  const [title, setTitle] = useState('');
  const [description, setDescription] = useState('');
  const [content, setContent] = useState('');
  const [img, setImg] = useState('');
  const [created, setCreated] = useState('');
  const [views, setViews] = useState(1);
  const [attractive, setAttractive] = useState(false);
  const [status, setStatus] = useState(false);
  const [error, setError] = useState(null);

  const { id } = useParams();
  const history = useHistory();
```

```
useEffect(() => {
  axios
    .get(`https://615c37884a360f0017a81496.mockapi.io/api/v1/news/${id}`)
    .then((response) => {
      setTitle(response.data.title);
      setDescription(response.data.description);
      setContent(response.data.content);
      setImg(response.data.img);
      setCreated(response.data.created);
      setViews(response.data.views);
      setAttractive(response.data.attractive);
      setStatus(response.data.status);
    })
    .catch((error) => {
      console.log(error);
      setError('Unable to fetch news item.');
    });
}, [id]);

const handleSubmit = (e) => {
  e.preventDefault();
  const updatedNews = {
    title,
    description,
    content,
    img,
    created,
    views,
```

```
    attractive,
    status,
  };
  axios
    .put(
      `https://615c37884a360f0017a81496.mockapi.io/api/v1/news/${id}`,
      updatedNews
    )
    .then(() => {
      history.push('/dashboard');
    })
    .catch((error) => {
      console.log(error);
      setError('Unable to update news item.');
    });
};

return (
  <div>
    <h1>Edit News Item</h1>
    {error && <div>{error}</div>}
    <form onSubmit={handleSubmit}>
      <div>
        <label htmlFor="title">Title:</label>
        <input
          type="text"
          id="title"
          value={title}
          onChange={(e) => setTitle(e.target.value)}
```

```
    required
   />
  </div>
  <div>
   <label htmlFor="description">Description:</label>
   <textarea
    id="description"
    value={description}
    onChange={(e) => setDescription(e.target.value)}
    required
   />
  </div>
  <div>
   <label htmlFor="content">Content:</label>
   <textarea
    id="content"
    value={content}
    onChange={(e) => setContent(e.target.value)}
    required
   />
  </div>
  <div>
   <label htmlFor="img">Image URL:</label>
   <input
    type="url"
    id="img"
    value={img}
    onChange={(e) => setImg(e.target.value)}
    required
```

/>