

## Bynry JOBS==> FrontEnd Task

**Objective:** To design a user interface where there must be specific dashboards for user and admins in such a way that user could able to view all the users present and admin could view and do certain changes like adding of values, removing of values and editing of existing of values

### **Requirements :**

The key functionalities of the application include:

1. **Profile Display:** Create a webpage that presents a collection of profiles, each comprising essential information such as the person's name, photograph, and a brief description.
2. **Interactive Mapping:** Incorporate an interactive map component that can dynamically display addresses based on user interactions. This map will allow users to see the geographical location associated with each profile.
3. **SummaryIntegration:** Implement a "Summary" button adjacent to each profile. Clicking this button should trigger the display of the map component with a marker indicating the precise address of the selected profile.
4. **MapServices Integration:** Utilise external map services like Google Maps or Mapbox To Integrate the mapping functionality into the application. This entails setting up markers and correctly rendering addresses on the map.
5. **User-Friendly Experience:** Ensure that the application offers a smooth and intuitive user experience, enabling users to easily navigate profiles and access mapped addresses without confusion.
6. **Profile Data Management :** Allow administrators to add, edit, or delete profiles.
7. This will require an admin panel or dashboard to manage the profile data efficiently.

8. Search and Filter Functionality : Provide users with the ability to search and filter profiles based on different criteria, such as name, location, or other attributes. This enhances the usability of the application.

9. Responsive Design : Ensure that the application is responsive and mobile-friendly so that users can access it from various devices, including smartphones and tablets.

10. Error Handling Implement robust error handling and validation mechanisms to handle issues gracefully, such as invalid addresses or failed map service requests.

11. Loading Indicators : Include loading indicators of progress bars to give users feedback when the application is fetching data or rendering the map.

12. Profile Details: Create a separate profile details view that provides more in-depth information about each profile when a user clicks on a profile card.

### **Step by Step working mechanism**

1)At first we have to install required tools like react and nodejs

2)After installing them type command `npx create-react-app ui` here ui is the file name taken by me

3)After installing we have to keep directory towards ui so in vscode(which is prerequisite) we open terminal in vscode and type `cd ui` and then we start react code by typing `npm start` and we see picture as below



Edit `src/App.js` and save to reload.

[Learn React](#)

4) We check the installed packages in node modules

5) First let us create two different login pages for login as user and login as admin

6) We use one of the core topics in React.js known as useState

useState is a **React Hook** that allows functional components to manage state. It enables components to maintain and update values over time without requiring a class-based approach.

7) We begin this by giving as below

```
const [email, setEmail] = useState("");
const [password, setPassword] = useState("");
const [error, setError] = useState("");
const [role, setRole] = useState(null);
const [loading, setLoading] = useState(false);
```

This is given like this because initial values are empty

8) We also used a loader page so that this UI looks pretty good and say for dummy authentication below are used

```
if (email === "admin@example.com" && password === "password" && role === "admin") {
    setLoading(true);
    alert("Admin Login Successful!");
    setTimeout(() => {
        setLoading(false);
        navigate("/admin");
    }, 3000);
} else if (email === "user@example.com" && password === "password" && role === "user") {
    setLoading(true);
    alert("User Login Successful!");
    setTimeout(() => {
        setLoading(false);
        navigate("/display");
    }, 5000);
} else {
    setError("Invalid email or password");
}
};
```

9) We use back button to return to previous page and it is achieved through ternary operator and also role is changed based on this as below

```
{role && <button className="back-button" onClick={() =>
setRole(null)}>Back</button>}
{!role ? (
    <>
        <h2>Select Login Type</h2>
        <div className="button-group">
            <button onClick={() => setRole("user")}>Login as
User</button>
            <button onClick={() => setRole("admin")}>Login as
Administrator</button>
        </div>
    </>
)
}
```

```

        </>
    ) : (
        <div className="login-container">
            <h2>Login as {role === "admin" ? "Administrator" :
"User"}</h2>
            {error && <p className="error">{error}</p>}
            <form onSubmit={handleSubmit}>
                <input
                    type="email"
                    placeholder="Email"
                    value={email}
                    onChange={(e) => setEmail(e.target.value)}
                />
                <input
                    type="password"
                    placeholder="Password"
                    value={password}
                    onChange={(e) => setPassword(e.target.value)}
                />
                <button type="submit">Login</button>
            </form>
        </div>
    )}
</div>
{loading && (
    <div className="overlay">
        <div className="loader"></div> {/* Circular loader */}
    </div>
)}

```

10)Based on above codes we get display as below

### Select Login Type

Login as User

Login as Administrator

11) If invalid credentials were given then we get as below

Back

### Login as User

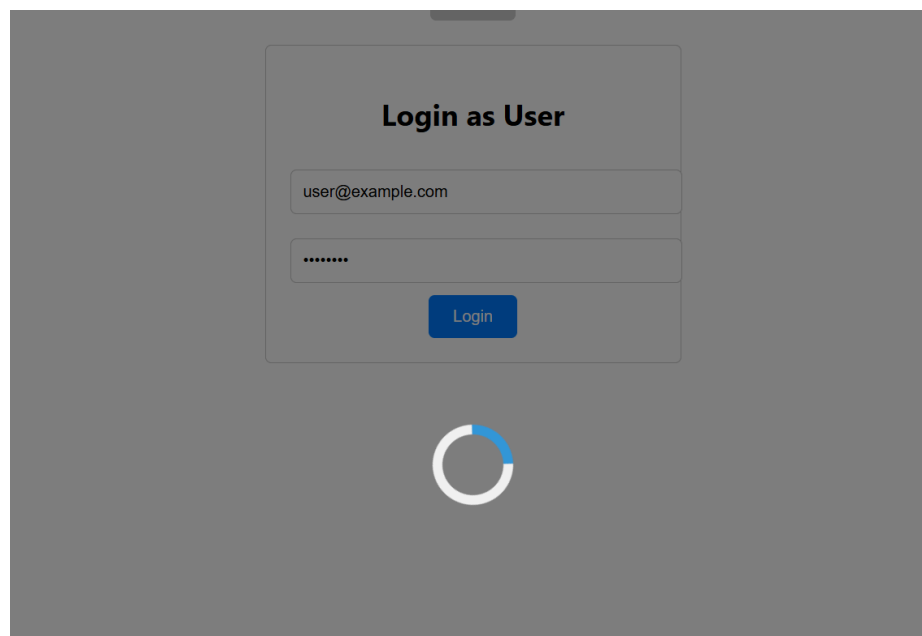
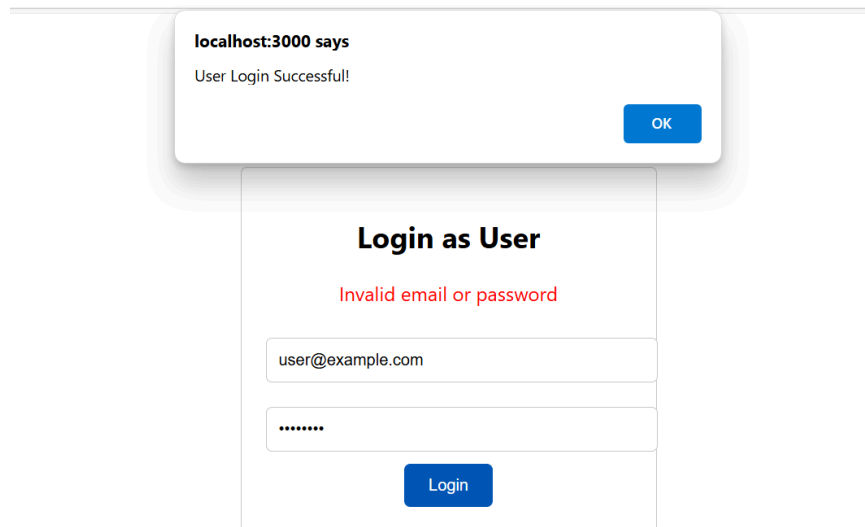
Invalid email or password

user@example.com

...

Login

12) If the valid credentials were given then we get alert button saying successful login and after that there will be a loading page



13) But we observe a difference in pages that display after login i.e for user login we see users only but login as admin we have authentication to add, edit and delete the existing users

## LOGIN THROUGH USER PAGE






[Logout](#)

Q

Name

▼

User Details

Name	Position	Email ID
 John Doe	HR	john@example.com
 Smith	Front-End	jane@example.com
 Sam Wilson	DBAdmin	sam@example.com
 David	Back-End	emily@example.com
 Michael Brown	QA-Intern	michael@example.com

14) We also observe there is a logout page which returns page to login page and also a search icon with filter values and table

15) The search icon was designed in such a way that based on user required filter the searching will be done and it also means we can't search those contents that are not present in this filter

[Logout](#)


Q

Name

▼

John

User Details

Name	Position	Email ID
 John Doe	HR	john@example.com



Logout

Q Name


User Details

Name	Position	Email ID
------	----------	----------

Logout

Q Position

User Details

Name	Position	Email ID
 John Doe	HR	john@example.com

16) We use `useState` and set default filter to name and also for display of input when clicking on search icon

```
const [searchVisible, setSearchVisible] = useState(false);
const [searchTerm, setSearchTerm] = useState("");
const [filterBy, setFilterBy] = useState("name");
```

Logic for searching

```
const filteredUsers = users.filter((user) => {
  return
  user[filterBy].toLowerCase().includes(searchTerm.toLowerCase());
});
```






Code for search visibility

```
const toggleSearch = () => {
  setSearchVisible(!searchVisible);
};
```

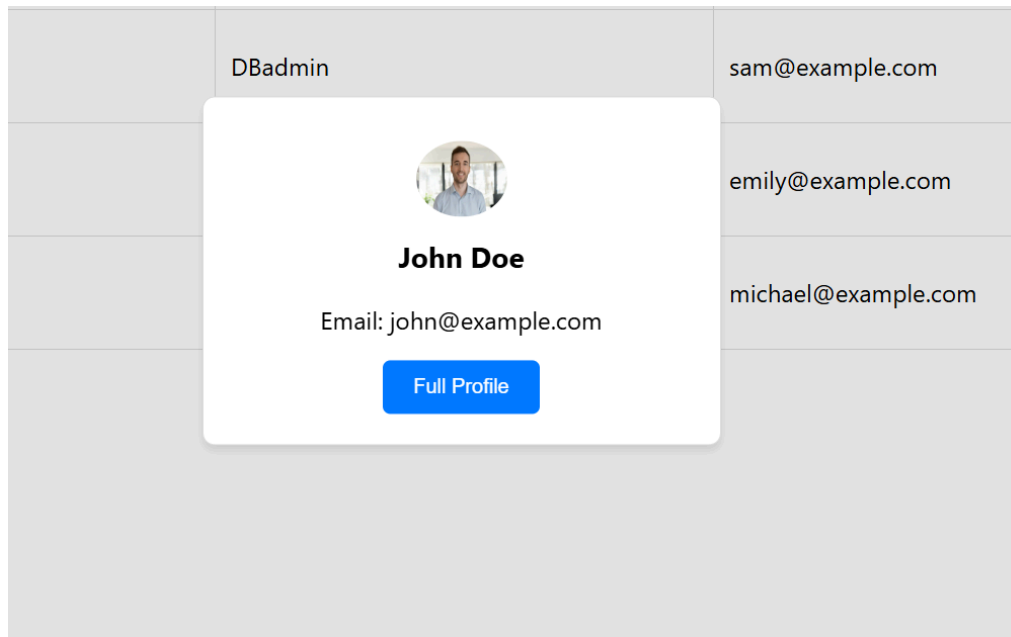
In html we write like this

```
<FaSearch className="search-icon" onClick={toggleSearch} />
  <select
    value={filterBy}
    onChange={ (e) => setFilterBy(e.target.value)}
    style={{ padding: '8px', marginRight: '10px' }}
  >
    <option value="name">Name</option>
    <option value="position">Position</option>
    <option value="email">Email</option>
    <option value="role">Role</option>
  </select>
  <input
    type="text"
    className="search-input"
    placeholder="Search..."
    value={searchTerm}
    onChange={ (e) => setSearchTerm(e.target.value)}
  />
```


17) Here the table displays photo of the person, name, mail id and position and also hovering on person name we see explore more means we can get much information on clicking his name

Name	Position	Email ID
 John Doe <div>Explore more</div>	HR	john@example.com
 Smith	Front-End	jane@example.com
 Sam Wilson	DBAdmin	sam@example.com
 David	Back-End	emily@example.com
 Michael Brown	QA-Intern	michael@example.com

18)On clicking name we get a card.The card is designed in such a way that it can be closed when we click outside the card and also the backgorund of the card is gray which is as shown below



19)On Clicking Full profile we need to get below way

  
**John Doe**  
Bio: Experienced HR professional  
Age: 35  
Phone: 123-456-7890  
Email: john@example.com

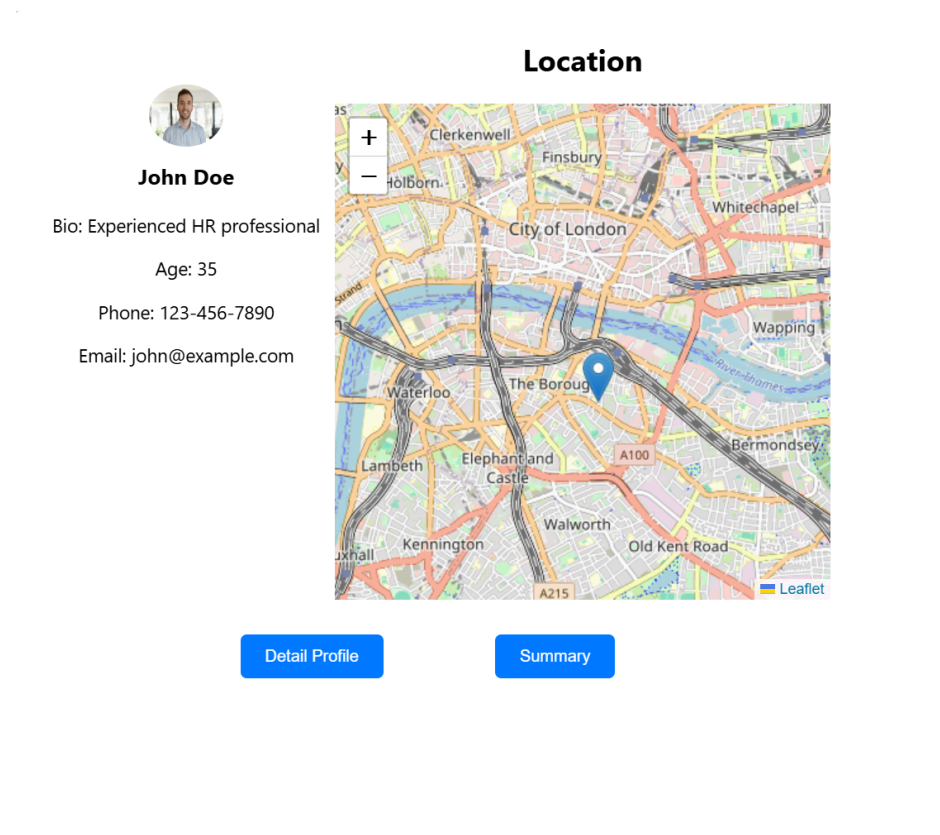
[Detail Profile](#)

[Summary](#)

20)In general react gives us error when there is no latitude and longitude so to avoid that i have used alert of js

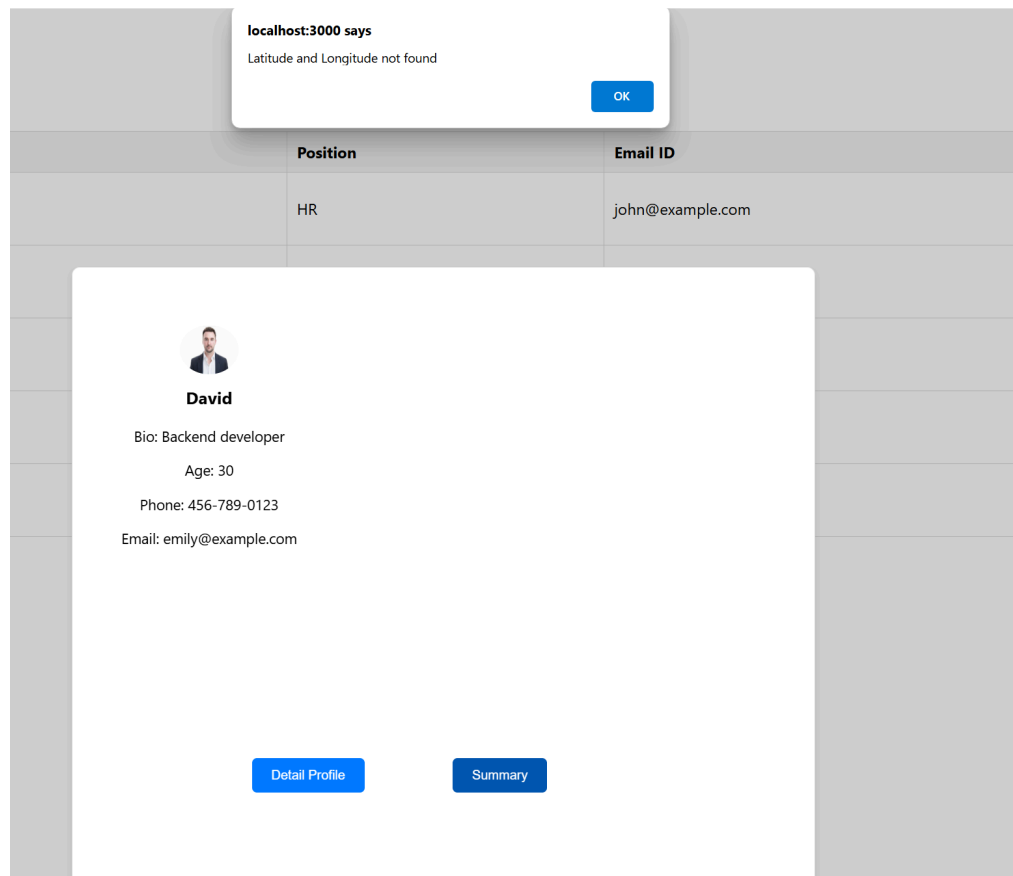
Following figures show you the difference between with lat and long and without it

```
{ name: "John Doe", role: "Admin", position: "HR", email: "john@example.com", bio: "Experienced HR professional", age: 35, phone: "123-456-7890", address: { lat: 51.505, lng: -0.09 },profileImage:"https://as2.ftcdn.net/v2/jpg/03/64/21/11/1000_F_364211147_1qgLVxv1Tcq00hz3FawUfrtONzz8nq3e.jpg" }
```



## Without Lat and long parameters

```
{ name: "David", role: "User", position: "Back-End", email: "emily@example.com", bio: "Backend developer", age: 30, phone: "456-789-0123", profileImage: "https://th.bing.com/th/id/OIP.lrODV181gEQh9tI4_Og2zwHaE8?rs=1&pid=ImgDetMain" },
```



## Function which checks for lat and long value

```
const handleSummaryClick = () => {  
  if (!fullProfile?.address?.lat || !fullProfile?.address?.lng) {  
    alert("Latitude and Longitude not found");  
    return;  
  }  
  setMapVisible(true);  
}
```

```
};
```

20) To implement map location based on lat and long we need to call google api and should use a library called react-leaflet

react-leaflet is a **React library** that provides components for integrating **Leaflet.js**, an open-source mapping library, into React applications. It allows developers to create **interactive maps** with markers, popups, layers, and other geospatial features.

## Basic usage in react

**MapContainer** – Creates the map with a specified center and zoom level.

**TileLayer** – Loads a map background from OpenStreetMap (or other providers).

**Marker** – Adds a location pin on the map.

**Popup** – Displays a message when clicking the marker.

Code for map display

```
<MapContainer
    center={ [fullProfile.address.lat,
fullProfile.address.lng] }
    zoom={13}
    scrollWheelZoom={false}
```

```

        style={{ width: "100%", height: "400px" }}

        >

        <TileLayer

url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"

        />

        <Marker

            position={[fullProfile.address.lat,
fullProfile.address.lng]}

            icon={new L.Icon({ iconUrl:
'https://unpkg.com/leaflet@1.7.1/dist/images/marker-icon.png' })}

        >

            <Popup>

                <h3>{fullProfile.name}</h3>

                <p>{fullProfile.role}</p>

                <p>{fullProfile.email}</p>

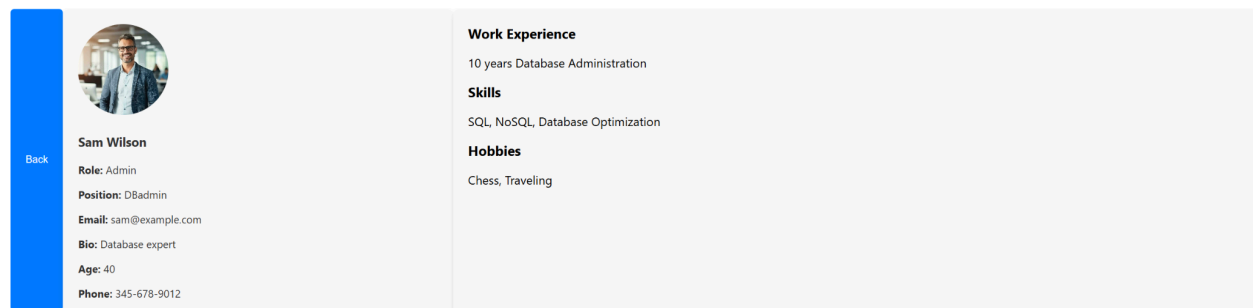
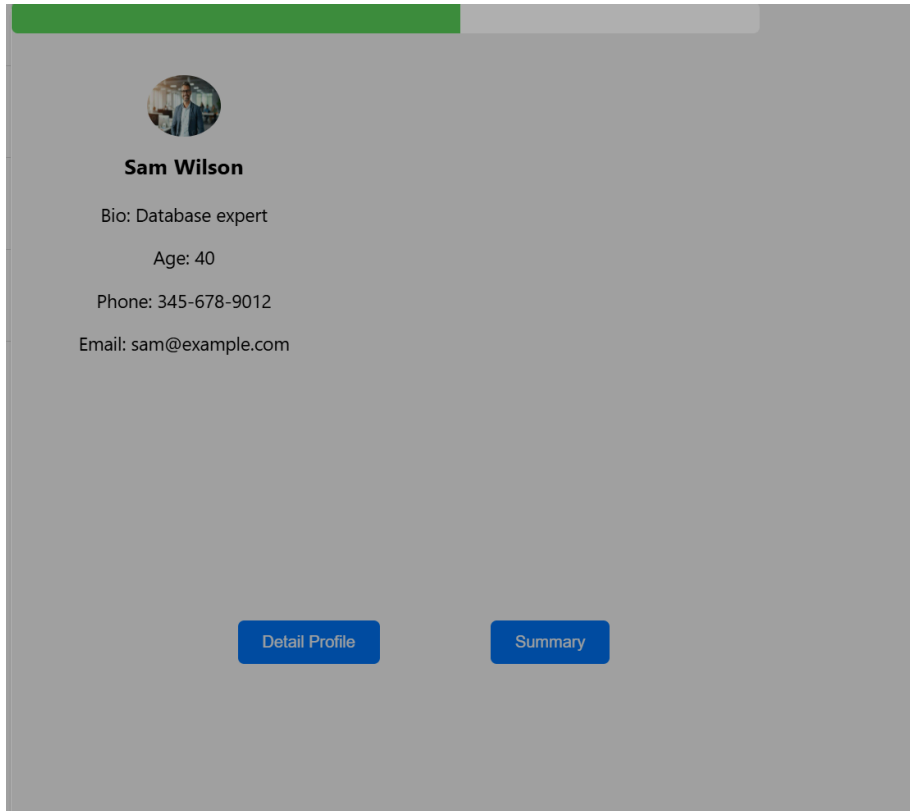
            </Popup>

        </Marker>

    </MapContainer>

```

21) There are two buttons namely Detail Profile and Summary. We know about Summary button but this Detail Profile displays about much data about profile. In this also we use loader namely progress bar and then detail profile page will be generated. The following figures show how they are displayed



The back button is useful to return to user page




# LOGIN THROUGH ADMIN PAGE

[Logout](#)

User Details

ID	Name	Location	Position	Bio	Actions		
1	John Doe	Scotland	HR	Experienced HR professional	ADD	EDIT	DELETE
2	Smith	London	Front-End	Passionate about UI/UX	ADD	EDIT	DELETE
3	Sam Wilson	Delhi	DBadmin	Database expert	ADD	EDIT	DELETE
4	David	Chicago	Back-End	Backend developer	ADD	EDIT	DELETE
5	Michael Brown	Ontario	QA-Intern	Aspiring QA engineer	ADD	EDIT	DELETE

22)The below page is displayed when we click on EDIT



Name:

Role:

Position:

Email:

Bio:

Age:

Phone:

Work Experience

Skills

Hobbies

Back

Submit

23)In order to navigate from page to page we use react library called navigator

The term "**Navigator**" in React generally refers to **navigation libraries** used for handling page transitions in React applications. Depending on whether you're working with a web or mobile app, the primary navigation libraries are:

1. **React Router** (for web apps)
2. **React Navigation** (for React Native mobile apps)

Code for implementing navigators

```
import logo from './logo.svg';

import './App.css';

import Login from "./components/login";

import Display from "./components/Display"

import { BrowserRouter as Router, Routes, Route } from "react-router-dom";

import Admindisplay from "./components/Admindisplay"

import Detail from "./components/Detail";

import Detaile from "./components/Detaile";

function App() {

  return (

    <Router>

    <Routes>

      <Route path="/" element={<Login />} />

      <Route path="/display" element={<Display />} />

      <Route path="/admin" element={<Admindisplay />} />

    </Routes>

    </Router>

  )
}
```

```

    <Route path="/detail/:id" element={<Detail />} />

    <Route path="/detaile/:id" element={<Detaile/>} />

  </Routes>

</Router>

);
}

export default App;

```

For getting previous page we use code like below

```

const navigate = useNavigate();

const handleLogout = () => {

  navigate("/");

};

```