Exercise: Create a "Profile Card" Component using JS and Create React App

Goal: Build a "Profile Card" React component that displays user details (name, age, and profile picture) passed as props using **JavaScript**.

---

1. Setup

Initialize the Project

Run the following commands to create a new React app using Create React App (no TypeScript):

```
npx create-react-app profile-card-app
cd profile-card-app
npm install
```

Start the Development Server

```
npm start
```

---

2. Component Creation

Create a `ProfileCard.js` File

Inside the `src/components/` folder, create the file `ProfileCard.js`:

`javascript`

```
import React from 'react';
import './ProfileCard.css';

const ProfileCard = ({ name, age, profilePicture, description }) => {
  return (
    <div className="profile-card">
      <img src={profilePicture} alt={`${name}'s profile`}
className="profile-picture" />
      <h2>{name}</h2>
      {age && <p>Age: {age}</p>}
      {description && <p>{description}</p>}
    </div>
  );
};

export default ProfileCard;
```

Create a `ProfileCard.css` File

In the same `src/components/` folder, create `ProfileCard.css` for optional styling:

`css`

```
.profile-card {
  border: 1px solid #ccc;
  border-radius: 10px;
  padding: 20px;
  text-align: center;
  width: 200px;
  margin: 10px auto;
  box-shadow: 0px 4px 6px rgba(0, 0, 0, 0.1);
}

.profile-picture {
  width: 100px;
  height: 100px;
  border-radius: 50%;
  margin-bottom: 15px;
}

h2 {
  margin: 10px 0;
}

p {
  color: #555;
}
```

---

3. Usage in `App.js`

Update the `src/App.js`

Replace the default content in `App.js` to import and render the `ProfileCard` component:

`javascript`

```
import React from 'react';
import './App.css'; // Optional global styles
import ProfileCard from './components/ProfileCard';

const App = () => {
  const profiles = [
    { name: 'John Doe', age: 30, profilePicture:
'https://via.placeholder.com/150', description: 'Software Engineer' },
    { name: 'Jane Smith', age: 25, profilePicture:
'https://via.placeholder.com/150', description: 'Graphic Designer' },
    { name: 'Alice Johnson', profilePicture:
'https://via.placeholder.com/150', description: 'Photographer' }, // No
age
```

```
  ];

  return (
    <div className="app">
      <h1>Profile Cards</h1>
      {profiles.map((profile, index) => (
        <ProfileCard
          key={index}
          name={profile.name}
          age={profile.age}
          profilePicture={profile.profilePicture}
          description={profile.description}
        />
      ))}
    </div>
  );
};

export default App;
```

---

4. Styling in `App.css` (Optional)

Add some global styles in `src/App.css`:

`css`

```
.app {
  text-align: center;
  padding: 20px;
}

h1 {
  font-size: 2rem;
  color: #333;
}
```

---

5. File Structure

After the changes, your file structure should look like this:

```
/profile-card-app
├── public/                  # Static assets
├── src/                     # Source files
│   ├── components/          # Component folder
│   │   ├── ProfileCard.js   # ProfileCard component
│   │   ├── ProfileCard.css  # Optional CSS for styling
│   ├── App.js               # Main application file
│   ├── App.css              # Optional global styles
│   ├── index.js             # React entry point
│   └── index.css            # Optional global styles
├── package.json             # Package manager file
```

```
├── node_modules/            # Installed dependencies
└── README.md                # Documentation
```

---

6. Run the App

Start the app with:

```
npm start
```

---

7. Output

You should see a "Profile Cards" page displaying multiple user cards with their profile pictures, names, ages (if provided), and descriptions (if provided).

Challenge (Optional Enhancements)

1. **Add More Props**:
   - Include fields like location, hobbies, or a link to social profiles.
2. **Improve Styling**:
   - Use a library like `Bootstrap`, `Material-UI`, or custom SCSS for advanced styling.
3. **Dynamic Profile Data**:
   - Fetch profiles from a REST API instead of hardcoding them. Use tools like `axios` or the Fetch API.

This JavaScript version simplifies the original TypeScript implementation while maintaining all functionality.