Register (Form + API)

JSON Server

🛠️ Step 6: Setup JSON Server

In your project root, create `db.json`:

```
{
  "users": [{
    "id":1,
    "username": "cj",
    "password":"1234"
}]
}
```

Run backend:

```
npx json-server --watch db.json --port 3001
```

✅ Final Project Structure

```
MyAuthApp/
│
├── App.js
├── db.json                # backend for json-server
│
├── screens/
│   ├── LoginScreen.js
│   ├── UserProfile.js     # Register + Search + Edit
│   └── UserList.js        # List + Delete
│
├── package.json
└── node_modules/
```

👉 Now you have **full CRUD** with Expo + JSON Server:

- **Login** → `LoginScreen.js`
- **Register/Search/Edit** → `UserProfile.js`
- **List/Delete** → `UserList.js`

Frontend with ReactNative

🛠️ Step 1: Create a new Expo app

```
npx create-expo-app --template
cd MyAuthApp
npx expo install react-dom react-native-web @expo/metro-runtime
npm run web
```

🛠️ Step 2: Install navigation

We'll use React Navigation:

```
npm install @react-navigation/native @react-navigation/stack
npm install react-native-screens react-native-safe-area-context
```

Modify **App.js** to test navigation:

```
import React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createStackNavigator } from '@react-navigation/stack';
import { View, Text } from 'react-native';

const Stack = createStackNavigator();

function Home() {
  return <View><Text>Hello Expo!</Text></View>;
}

export default function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator>
        <Stack.Screen name="Home" component={Home} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}
```

Run app: You should see a screen with “Hello Expo!”.

Step 3: Add LoginScreen

Step 3.1: Create a blank Login screen

Create `screens/LoginScreen.js` :

```
import React from 'react';
import { View, Text } from 'react-native';

export default function LoginScreen() {
  return (
    <View>
      <Text>Login Screen</Text>
    </View>
  );
}
```

Update **App.js** to use it:

```
import LoginScreen from './screens/LoginScreen';

<Stack.Screen name="Login" component={LoginScreen} />
```

Step 3.2: Add username & password inputs

Update `LoginScreen.js` :

```jsx
import React, { useState } from 'react';
import { View, TextInput } from 'react-native';

export default function LoginScreen() {
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');

  return (
    <View style={{ padding:20 }}>
      <TextInput
        placeholder="Username"
        value={username}
        onChangeText={setUsername}
        style={{ borderWidth:1, marginBottom:10, padding:8 }}
      />
      <TextInput
        placeholder="Password"
        secureTextEntry
        value={password}
        onChangeText={setPassword}
        style={{ borderWidth:1, marginBottom:10, padding:8 }}
      />
    </View>
  );
}
```

✅ Run: You can type into the fields.

Step 3.3: Add Login button (no API yet)

```
import React, { useState } from 'react';
import { View, TextInput, Button } from 'react-native';

export default function LoginScreen() {
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');

  const handleLogin = () => {
    alert('Login pressed: ' + username);
  };

  return (
    <View style={{ padding:20 }}>
      <TextInput placeholder="Username" value={username}
onChangeText={setUsername} style={{ borderWidth:1, marginBottom:10,
padding:8 }}/>
      <TextInput placeholder="Password" secureTextEntry value=
{password} onChangeText={setPassword} style={{ borderWidth:1,
marginBottom:10, padding:8 }}/>
      <Button title="Login" onPress={handleLogin} />
    </View>
  );
}
```

✅ Run: Pressing **Login** shows an alert with username.

Step 3.4: Connect to JSON Server API

Final step → check credentials from `db.json` .

```
import React, { useState } from 'react';
import { View, TextInput, Button } from 'react-native';
import { useNavigation } from '@react-navigation/native';

export default function LoginScreen({ navigation }) {
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');
    const navigation = useNavigation();

  const handleLogin = () => {
    let url = `http://localhost:3001/users?
username=${username}&password=${password}`;
    fetch(url)
      .then(res => res.json())
      .then(data => {
        if (data.length > 0) {
          navigation.navigate('UserProfile');
        }
      });
  };

  return (
    <View style={{ padding:20 }}>
      <TextInput placeholder="Username" value={username}
onChangeText={setUsername} style={{ borderWidth:1, marginBottom:10,
padding:8 }}/>
      <TextInput placeholder="Password" secureTextEntry value=
{password} onChangeText={setPassword} style={{ borderWidth:1,
marginBottom:10, padding:8 }}/>
      <Button title="Login" onPress={handleLogin} />
    </View>
  );
}
```

✅ Run:

- Type `cj / 1234` (if exists in `db.json`).
- Login → navigates to **UserProfile**.

🛠️ Step 4: Add UserProfile.js (Register + Search + Edit merged)

🛠️ Step 4.1: Blank UserProfile screen

`screens/UserProfile.js`

```
import React from 'react';
import { View, Text } from 'react-native';

export default function UserProfile() {
  return (
    <View>
      <Text>User Profile Screen</Text>
    </View>
  );
}
```

✅ Run: Navigate to **UserProfile**, you'll see simple text.

🛠️ Step 4.2: Add input fields

Add form inputs for username, password, email, fullname.

```
import React, { useState } from 'react';
import { View, TextInput } from 'react-native';

export default function UserProfile() {
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');
  const [email, setEmail] = useState('');
  const [fullname, setFullname] = useState('');

  return (
    <View style={{ padding:20 }}>
      <TextInput placeholder="Username" value={username}
onChangeText={setUsername} style={{ borderWidth:1, marginBottom:10,
padding:8 }}/>
      <TextInput placeholder="Password" secureTextEntry value=
{password} onChangeText={setPassword} style={{ borderWidth:1,
marginBottom:10, padding:8 }}/>
      <TextInput placeholder="Email" value={email} onChangeText=
{setEmail} style={{ borderWidth:1, marginBottom:10, padding:8 }}/>
      <TextInput placeholder="Full Name" value={fullname}
onChangeText={setFullname} style={{ borderWidth:1, marginBottom:10,
padding:8 }}/>
    </View>
  );
}
```

✅ Run: You can type into all fields.

🛠️ Step 4.3: Add buttons (no API yet)

Add **Search**, **Register**, and **Update** buttons.

```
import React, { useState } from 'react';
import { View, TextInput, Button } from 'react-native';

export default function UserProfile() {
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');
  const [email, setEmail] = useState('');
  const [fullname, setFullname] = useState('');

  return (
    <View style={{ padding:20 }}>
      <TextInput placeholder="Username" value={username}
onChangeText={setUsername} style={{ borderWidth:1, marginBottom:10,
padding:8 }}/>
      <TextInput placeholder="Password" secureTextEntry value=
{password} onChangeText={setPassword} style={{ borderWidth:1,
marginBottom:10, padding:8 }}/>
      <TextInput placeholder="Email" value={email} onChangeText=
{setEmail} style={{ borderWidth:1, marginBottom:10, padding:8 }}/>
      <TextInput placeholder="Full Name" value={fullname}
onChangeText={setFullname} style={{ borderWidth:1, marginBottom:10,
padding:8 }}/>

      <Button title="Search" onPress={() => alert('Search
pressed')} />
      <Button title="Register" onPress={() => alert('Register
pressed')} />
      <Button title="Update" onPress={() => alert('Update
pressed')} />
    </View>
  );
}
```

✅ Run: Pressing buttons shows alerts.

🛠️ Step 4.4: Add Search API (GET)

Implement **search by username**.

```
const handleSearch = () => {
  fetch(`http://localhost:3001/users?username=${username}`)
    .then(res => res.json())
    .then(data => {
      if (data.length > 0) {
        let u = data[0];
        setPassword(u.password);
        setEmail(u.email);
        setFullname(u.fullname);
      } else {
        alert('No user found');
      }
    });
};
```

Update Search button:

```
<Button title="Search" onPress={handleSearch} />
```

✅ Run: Typing a username (e.g., `cj`) and pressing **Search** loads data from `db.json`.

🛠️ Step 4.5: Add Register API (POST)

```
const handleRegister = () => {
  fetch('http://localhost:3001/users', {
    method:'POST',
    headers:{'Content-Type':'application/json'},
    body: JSON.stringify({ username, password, email, fullname })
  })
  .then(res => res.json())
  .then(data => alert('User created: ' + data.username));
};
```

Update Register button:

```
<Button title="Register" onPress={handleRegister} />
```

✅ Run: Fill fields → Register → New user saved in `db.json`.

🛠️ Step 4.6: Add Update API (PATCH)

We need the user's **id**. Modify Search to also set `id` :

```
const [id, setId] = useState(null);

const handleSearch = () => {
  fetch(`http://localhost:3001/users?username=${username}`)
    .then(res => res.json())
    .then(data => {
      if (data.length > 0) {
        let u = data[0];
        setId(u.id);
        setPassword(u.password);
        setEmail(u.email);
        setFullname(u.fullname);
        alert('User found: ' + u.username);
      } else {
        alert('No user found');
      }
    });
};
```

Now add Update function:

```
const handleUpdate = () => {
  if (!id) return alert('Search first!');
  fetch(`http://localhost:3001/users/${id}`, {
    method:'PATCH',
    headers:{'Content-Type':'application/json'},
    body: JSON.stringify({ username, password, email, fullname })
  })
  .then(() => alert('User updated!'));
};
```

Update button:

```
<Button title="Update" onPress={handleUpdate} />
```

✅ Run: Search user → change fields → Update → changes saved in `db.json` .

🎯 Now your `UserProfile.js` supports **Search, Register, Update** step by step.

🛠️ Step 5: Add UserList.js (Read + Delete)

🛠️ Step 5.1: Blank UserList screen

`screens/UserList.js`

```
import React from 'react';
import { View, Text } from 'react-native';

export default function UserList() {
  return (
    <View>
      <Text>User List Screen</Text>
    </View>
  );
}
```

✅ Run: Navigate to **UserList**, you see simple text.

🛠️ Step 5.2: Fetch and show raw list (no styling)

```
import React, { useEffect, useState } from 'react';
import { View, Text } from 'react-native';

export default function UserList() {
  const [users, setUsers] = useState([]);

  useEffect(() => {
    fetch('http://localhost:3001/users')
      .then(res => res.json())
      .then(setUsers);
  }, []);

  return (
    <View style={{ padding:20 }}>
      {users.map(u => (
        <Text key={u.id}>{u.username}</Text>
      ))}
    </View>
  );
}
```

✅ Run: Shows all usernames from `db.json` .

🛠️ Step 5.3: Use FlatList for cleaner rendering

```
import React, { useEffect, useState } from 'react';
import { View, Text, FlatList } from 'react-native';

export default function UserList() {
  const [users, setUsers] = useState([]);

  useEffect(() => {
    fetch('http://localhost:3001/users')
      .then(res => res.json())
      .then(setUsers);
  }, []);

  return (
    <View style={{ flex:1, padding:20 }}>
      <FlatList
        data={users}
        keyExtractor={(item) => item.id.toString()}
        renderItem={({ item }) => (
          <Text>{item.username} ({item.email})</Text>
        )}
      />
    </View>
  );
}
```

✅ Run: List shows `username (email)` .

🛠️ Step 5.4: Add Delete button (per user)

```
import React, { useEffect, useState } from 'react';
import { View, Text, Button, FlatList } from 'react-native';

export default function UserList() {
  const [users, setUsers] = useState([]);

  const loadUsers = () => {
    fetch('http://localhost:3001/users')
      .then(res => res.json())
      .then(setUsers);
  };

  useEffect(() => { loadUsers(); }, []);

  const handleDelete = (id) => {
    fetch(`http://localhost:3001/users/${id}`, { method:'DELETE' })
      .then(() => loadUsers());
  };

  return (
    <View style={{ flex:1, padding:20 }}>
      <FlatList
        data={users}
        keyExtractor={(item) => item.id.toString()}
        renderItem={({ item }) => (
          <View style={{ marginBottom:10 }}>
            <Text>{item.username} ({item.email})</Text>
            <Button title="Delete" onPress={() =>
handleDelete(item.id)} />
          </View>
        )}
      />
    </View>
  );
}
```

✅ Run: Tap **Delete**, user is removed from `db.json` and list refreshes.

🛠️ Step 5.5: (Optional) Add refresh button

If you want manual refresh:

```
<Button title="Reload" onPress={loadUsers} />
```

🎯 Now your `UserList.js` supports:

- Blank screen → list users → delete users.