React Frontend Learning Guide (5 Weeks)

PFA: Links to all free resources and materials are included on the last page of this PDF.

Note: Focus on functionality over styling.

Week 1: Routing, JSON, Signup/Login

- Learn routing using react-router-dom.

- Understand JSON methods like parse, stringify, setItem, getItem.

- Create login and signup forms that navigate to each other.

- On signup, store credentials and unique userId in localStorage.

- On login, validate email-password for the same user.

- Show success alert or error if mismatch.

Routing and Navigation

```
npm install react-router-dom
```

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

<Routes>
  <Route path="/login" element={<Login />} />
  <Route path="/signup" element={<Signup />} />
</Routes>
```

LocalStorage Signup Example

```
const users = JSON.parse(localStorage.getItem('users')) || [];
const userId = Date.now();
users.push({ email, password, userId });
localStorage.setItem('users', JSON.stringify(users));
```

LocalStorage Login Example

```
const user = users.find(
  (u) => u.email === email && u.password === password
);
if (user) {
  alert("Login Successful");
  sessionStorage.setItem("activeUser", JSON.stringify(user));
} else {
  alert("Invalid credentials");
}
```

Form Validation with Formik & Yup

- Create validation schema using Yup.

- Use Formik for form handling.

- Add validations to login and signup forms.

Validation Schema Example

```
import * as Yup from 'yup';

const validationSchema = Yup.object({
  email: Yup.string().email('Invalid Email').required('Required'),
  password: Yup.string().min(4, 'Min 4 characters').required('Required'),
});
```

Formik Usage Example

```
<Formik
  initialValues={{ email: '', password: '' }}
  validationSchema={validationSchema}
  onSubmit={(values) => console.log(values)}
>
  {({ handleChange, handleBlur }) => (
    <form>
      <input name="email" onChange={handleChange} onBlur={handleBlur} />
      ...
    </form>
  )}
</Formik>
```

Week 2: Navigation, Session, Protected Routing

- Redirect to /home after login.

- Create Navbar with links and Logout.

- Create /home, /products, /category pages.

- Use sessionStorage for session management.

- Protect routes using a wrapper component.

- Logout should clear session.

Protected Route Example

```
const ProtectedRoute = ({ children }) => {
  const isLoggedIn = sessionStorage.getItem("activeUser");
  return isLoggedIn ? children : <Navigate to="/login" />;
};
```

Logout Function

```
const logout = () => {
  sessionStorage.clear();
  navigate("/login");
};
```

Week 3: Category & Product Tables

- Create tables and add forms for product/category.

- Category: Name + Status (dropdown).

- Product: Name + dropdown of existing categories from localStorage.

- Save entries in localStorage by userId.

Category Dropdown

```
<select name="status">
  <option value="true">Yes</option>
  <option value="false">No</option>
</select>
```

Product Category Dropdown (map from localStorage)

```
const categories = JSON.parse(localStorage.getItem("categories")) || [];
<select name="category">
  {categories.map((cat) => (
    <option key={cat.id}>{cat.name}</option>
  ))}
</select>
```

Week 4: Edit & Delete Rows

- Add Edit and Delete buttons to each row.

- Use same form for editing with prefilled data.

- Update button replaces entry in localStorage.

- Delete removes entry from UI and localStorage.

Edit Logic

```
const handleEdit = (item) => {
  setFormData(item);
  setIsEdit(true);
};
```

Update Logic

```
const updatedItems = items.map((i) =>
  i.id === formData.id ? formData : i
);
localStorage.setItem("products", JSON.stringify(updatedItems));
```

Delete Logic

```
const newList = items.filter((i) => i.id !== id);
localStorage.setItem("products", JSON.stringify(newList));
```

Week 5: API Integration with Fetch

- Practice GET, POST, PUT, DELETE using JSONPlaceholder API.

- Setup apiService.js.

- Fetch data and display in table format.

- Use .then(res => res.json()).then(data => ...)

- Monitor requests in DevTools > Network.

- Always refetch after data mutation.

GET Example

```
fetch("https://jsonplaceholder.typicode.com/posts")
  .then((res) => res.json())
  .then((data) => console.log(data));
```

POST Example

```
fetch("https://jsonplaceholder.typicode.com/posts", {
  method: "POST",
  body: JSON.stringify({
    title: "foo",
    body: "bar",
    userId: 1
  }),
  headers: {
    "Content-type": "application/json; charset=UTF-8"
  }
});
```

PUT Example

```
fetch("https://jsonplaceholder.typicode.com/posts/1", {
  method: "PUT",
  body: JSON.stringify({
    id: 1,
    title: "updated",
    body: "changed body",
    userId: 1
  }),
  headers: {
    "Content-type": "application/json; charset=UTF-8"
  }
});
```

DELETE Example

```
fetch("https://jsonplaceholder.typicode.com/posts/1", {
  method: "DELETE"
});
```

Feel free to reach out on izaid.dev@gmail.com in case of doubts or clarifications.

PDF Last Updated On: 2025-07-16 11:34:16 UTC

References

1. https://reactjs.org/docs/getting-started.html
2. https://reactjs.org/docs/state-and-lifecycle.html
3. https://www.w3schools.com/js/js_storage.asp
4. https://www.w3schools.com/jsref/prop_win_sessionstorage.asp
5. https://formik.org/docs/overview
6. https://github.com/jquense/yup
7. https://jsonplaceholder.typicode.com/
8. https://www.w3schools.com/js/js_api_fetch.asp
9. https://jsonplaceholder.typicode.com/guide/#post
10. https://jsonplaceholder.typicode.com/guide/#put-patch
11. https://jsonplaceholder.typicode.com/guide/#delete
12. https://jsonplaceholder.typicode.com/todos/1