ToDo Application:

Created Re-usable View Modules using React and JavaScript XML.

Employed React Route, Navigate, Link functions.

Used Session storage to store username as Key.

Created a functionality so that only Authenticated Users can access specific URLs.

Used Map function for iterating Lists in React.

Used Axios Framework to call HTTP methods.

Used Postman for REST Api testing.

Used Formik library in React to create forms.

Preventing CORS Error:

@CrossOrigin(origins = “http://localhost:4200”)

Cannot display an Object or JSON directly in React from Back-end.

RETREIVE all todos for a User:  
 GET /users/{user\_name}/todos

Delete a todo:

DELETE /users/{user\_name}/ todos/{todo\_id}

Update a todo:  
 PUT /users/{user\_name}/todos/{todo\_id}

Create a todo:

POST /users/{user\_name}/todos

Only GETMapping is supported in browser.

React:

Whenever something in the component changes, or the state of the component changes, the view has to be updated for it, render() will be called.

When the component is loaded for the first time and shown on the browser. This process of actually putting the component on the browser is called Mounting.

API call should not be done in Constructor, because state will not be initialized until the API call is completed.

Always call an API from ComponentDidMount():  
Component life cycle method

Life Cycle of React Component:

First Constructor -> render -> ComponentDidMount()

In ComponentDidMOunt; is called immediately after the component is mounted.

state is updated. React again calls render()

shouldComponentUpdate(): It is called before the render is called; it determines whether the change in properties, state should trigger a re-render.

Improves the performance.

If return false; it does not call render.

Render(), change in state; render is called.

MySQL:

Download MySQL WorkBench

MySQL Installer