Build a UI application Using ReactJS

This Lab provides a step-by-step guide for building and configuring a User interface application using ReactJS for Intern business application created in previous section.

Prerequisites:

- NPM
- Visual Studio
- Spring Boot Rest application(InternBusiness Application) or
- JAX-RS InternBusiness application
- MongoDB Compass Community(for Testing)

Business Scenario:

An application has to be developed for management of interns hired by Yash technologies. The application will provide following services:

1. Based on technical evaluation, intern will be hired and details will be entered in system. Interns technical level will be decided based on college semester marks. Interns will have to go through training based on levels and clear certifications.
2. Intern's details can be retrieved by HR to assign projects to intern.
3. Intern's personal details can be updated.
4. Intern's level can be updated based on semester scores.
5. Intern's details will be removed from system if intern has completed internship.

There will be two applications to suffice need of above system.

InternsUIApp will be developed using ReactJS and InternsBusinessApp will be business application which we have developed in previous sections.

ReactJS is JavaScript library used for building reusable UI components. According to React official documentation, following is the definition −

React is a library for building compassable user interfaces. It encourages the creation of reusable UI components, which present data that changes over time. Lots of people use React as the V in MVC. React abstracts away the DOM from you, offering a simpler programming model and better performance. React can also render on the server using Node, and it can power native apps using React Native. React implements one-way reactive data flow, which reduces the boilerplate and is easier to reason about than traditional data binding.

React Features

- **JSX** − JSX is JavaScript syntax extension. It isn't necessary to use JSX in React development, but it is recommended.
- **Components** − React is all about components. You need to think of everything as a component. This will help you maintain the code when working on larger scale projects.
- **Unidirectional data flow and Flux** − React implements one-way data flow which makes it easy to reason about your app. Flux is a pattern that helps keeping your data unidirectional.
- **License** − React is licensed under the Facebook Inc. Documentation is licensed under CC BY 4.0.

React Advantages

- Uses virtual DOM which is a JavaScript object. This will improve apps performance, since JavaScript virtual DOM is faster than the regular DOM.
- Can be used on client and server side as well as with other frameworks.
- Component and data patterns improve readability, which helps to maintain larger apps.

React Limitations

- Covers only the view layer of the app, hence you still need to choose other technologies to get a complete tooling set for development.
- Uses inline templating and JSX, which might seem awkward to some developers.

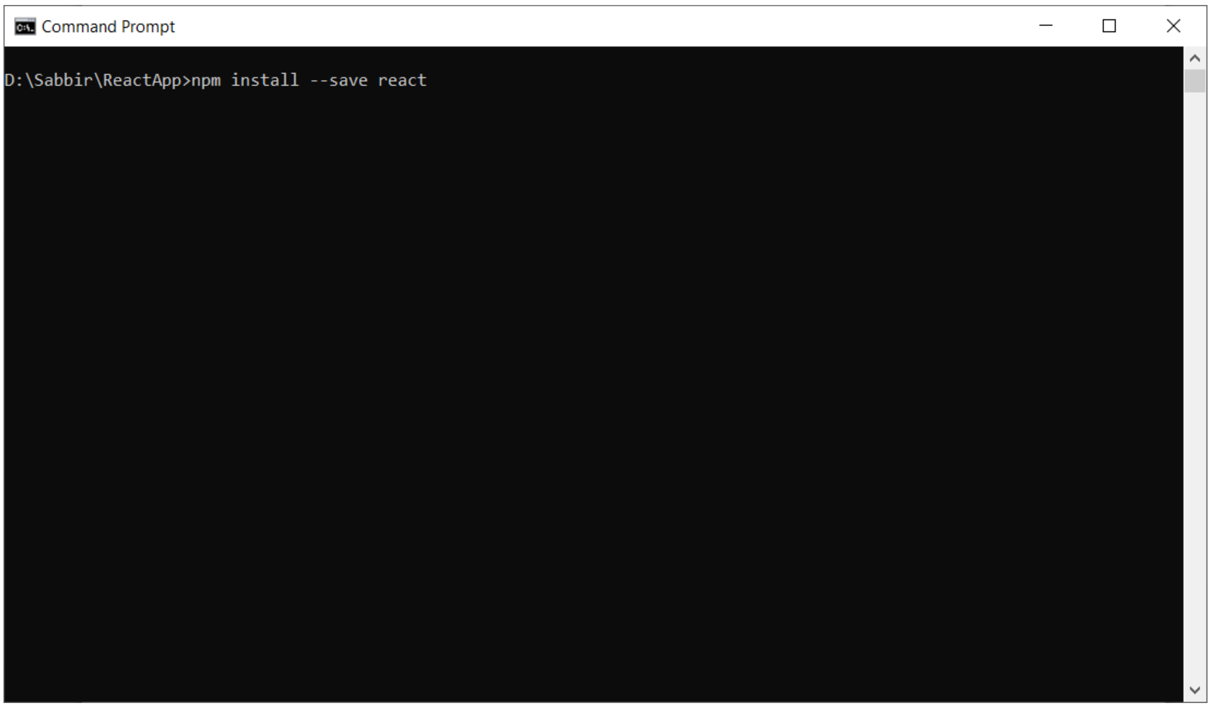
We will build basic UI Application using latest version of React and Visual studio code.

Create a folder in D:

D:\Sabbir\ReactApp

cd d:\Sabbir\ReactApp

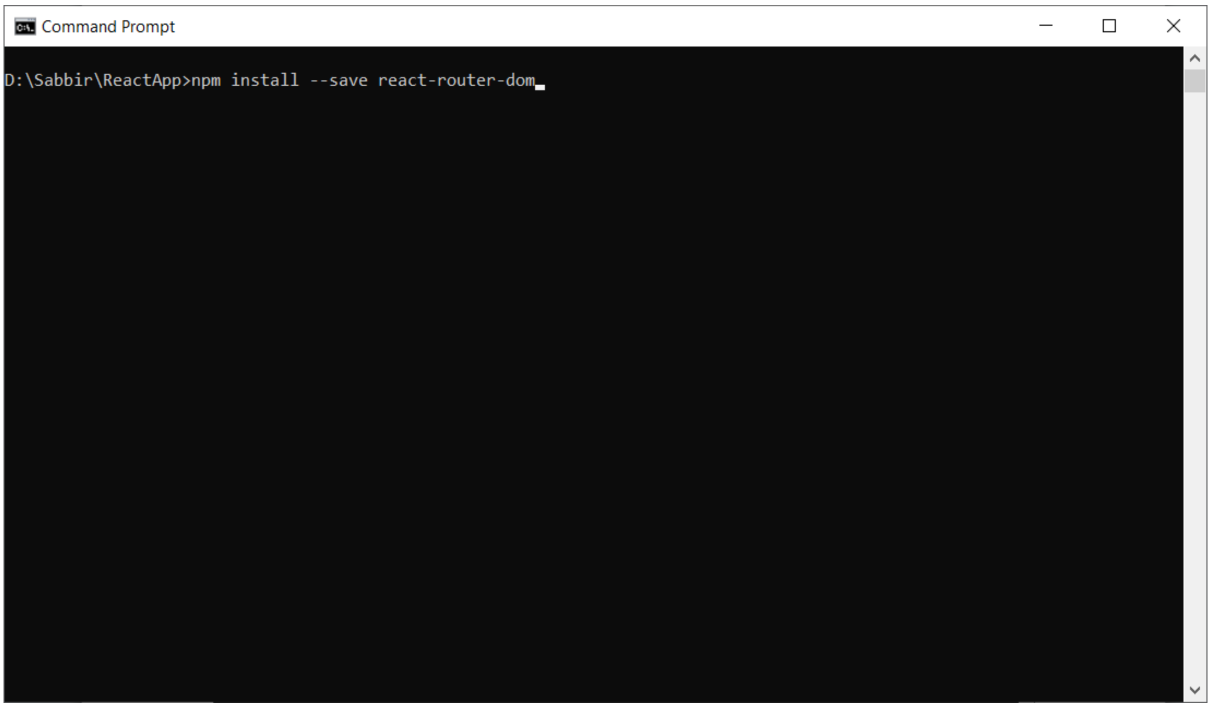
Type: `npm install –save react`



```
D:\Sabbir\ReactApp>npm install --save react
```

After above command,

Type: `npm install –save react-router-dom`



```
D:\Sabbir\ReactApp>npm install --save react-router-dom
```

Create React App

The create-react-app is an officially supported way to create React applications.

If you have NPM and Node.js installed, you can create a React application by first installing the create-react-app.

Install create-react-app by running this command in your terminal:

```
npm install create-react-app
```

Create a react application,

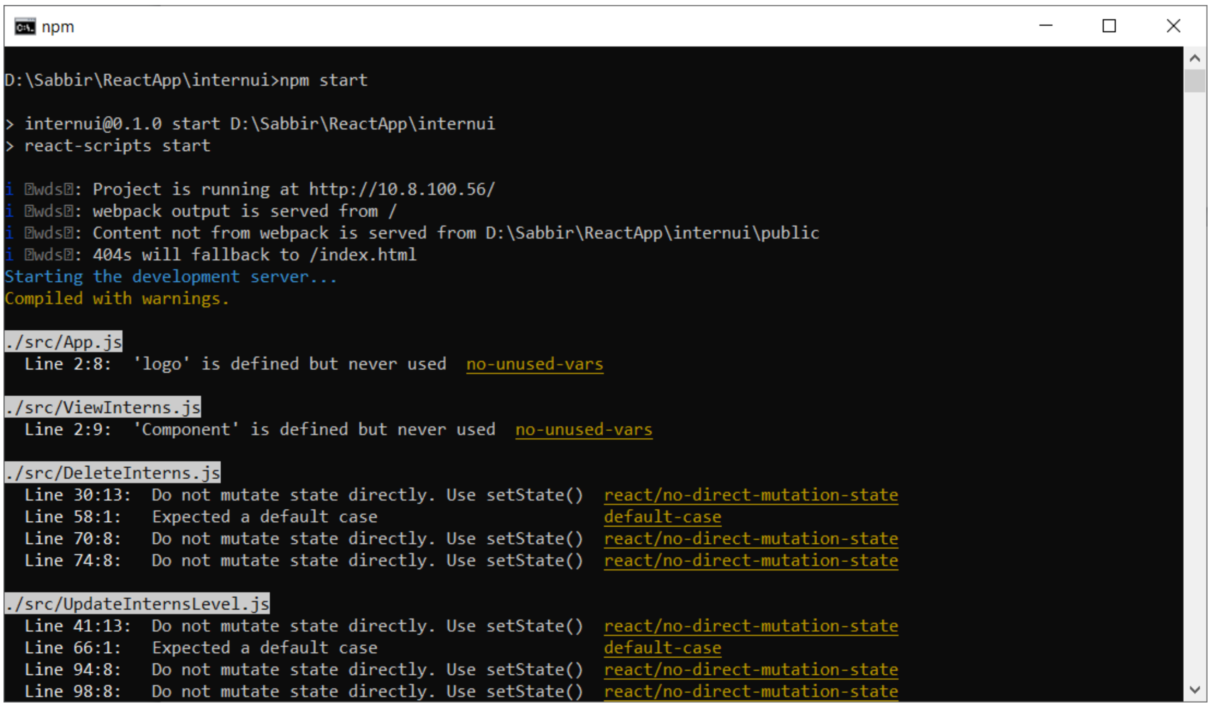
```
npx create-react-app internui
```

Run the React Application

If you followed the two commands above, you are ready to run your first *real* React application!

Run this command to move to the **internui** directory:

```
npm start
```



```
D:\Sabbir\ReactApp\internui>npm start

> internui@0.1.0 start D:\Sabbir\ReactApp\internui
> react-scripts start

i ｢wds｣: Project is running at http://10.8.100.56/
i ｢wds｣: webpack output is served from /
i ｢wds｣: Content not from webpack is served from D:\Sabbir\ReactApp\internui\public
i ｢wds｣: 404s will fallback to /index.html
Starting the development server...
Compiled with warnings.

./src/App.js
  Line 2:8:   'logo' is defined but never used  no-unused-vars

./src/ViewInterns.js
  Line 2:9:   'Component' is defined but never used  no-unused-vars

./src/DeleteInterns.js
  Line 30:13:  Do not mutate state directly. Use setState()  react/no-direct-mutation-state
  Line 58:1:   Expected a default case                       default-case
  Line 70:8:   Do not mutate state directly. Use setState()  react/no-direct-mutation-state
  Line 74:8:   Do not mutate state directly. Use setState()  react/no-direct-mutation-state

./src/UpdateInternsLevel.js
  Line 41:13:  Do not mutate state directly. Use setState()  react/no-direct-mutation-state
  Line 66:1:   Expected a default case                       default-case
  Line 94:8:   Do not mutate state directly. Use setState()  react/no-direct-mutation-state
  Line 98:8:   Do not mutate state directly. Use setState()  react/no-direct-mutation-state
```

Open folder in Visual Studio code

In index.html mention below to add link for Bootstrap. It can also be installed using NPM

```
        <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css
" integrity="sha384-
MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"
crossorigin="anonymous">
```

To view intern data create a file ViewInterns.js in src folder

Introducing JSX

JSX is an XML/HTML-like syntax used by React that extends ECMAScript so that XML/HTML-like text can co-exist with JavaScript/React code. The syntax is intended to be used by preprocessors (i.e., transpilers like Babel) to transform HTML-like text found in JavaScript files into standard JavaScript objects that a JavaScript engine will parse.

Basically, by using JSX you can write concise HTML/XML-like structures (e.g., DOM like tree structures) in the same file as you write JavaScript code, then Babel will transform these expressions into actual JavaScript code. Unlike the past, instead of putting JavaScript into HTML, JSX allows us to put HTML into JavaScript.

By using JSX one can write the following JSX/JavaScript code:

```
var nav = (
    <ul id="nav">
      <li><a href="#">View Interns</a></li>
      <li><a href="#">Register Interns</a></li>
      <li><a href="#">Update Interns</a></li>
      <li><a href="#">Delete Interns</a></li>
    </ul>
);
```

And Babel a JavaScript compiler will transform it into this:

```
var nav = React.createElement(
   "ul",
   { id: "nav" },
   React.createElement(
      "li",
      null,
      React.createElement(
         "a",
         { href: "#" },
         "View Interns"
      )
   ),
   React.createElement(
      "li",
      null,
      React.createElement(
         "a",
         { href: "#" },
         "Register Interns"
      )
   ),
   React.createElement(
      "li",
      null,
      React.createElement(
         "a",
         { href: "#" },
         "Update Interns"
      )
   ),
   React.createElement(
      "li",
      null,
      React.createElement(
         "a",
         { href: "#" },
         "Delete Interns"
      )
   )
);
```

You can think of JSX as a shorthand for calling `React.createElement()`.

This is a reason why we need to import React in our component.

```
Import React from 'react';
```

What Are Components?

Components are self-sustaining, independent micro-entities that describe a part of your UI. An application's UI can be split up into smaller components where each component has its own code, structure, and API.

Facebook, for instance, has thousands of pieces of functionality interfaced together when you view their web application. Here is an interesting fact: Facebook comprises 30,000 components, and the number is growing. The component architecture allows you to think of each piece in isolation. Each component can update everything in its scope without being concerned about how it affects other components.

If we take Facebook's UI as an example, the search bar would be a good candidate for a component. Facebook's Newsfeed would make another component (or a component that hosts many sub-components). All the methods and AJAX calls concerned with the search bar would be within that component.

Components are also reusable. If you need the same component in multiple places, that's easy. With the help of JSX syntax, you can declare your components wherever you want them to appear, and that's it.

There are mainly two components in React:

- Functional Components
- Class Components

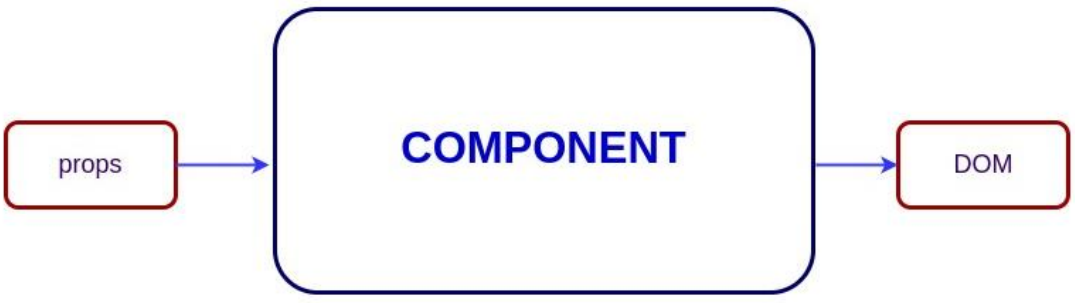
Props and State

Components need data to work with. There are two different ways that you can combine components and data: either as **props** or **state**. props and state determine what a component renders and how it behaves.

Let's start with props.

Props

If components were plain JavaScript functions, then props would be the function input. Going by that analogy, a component accepts an input (what we call props), processes it, and then renders some JSX code.



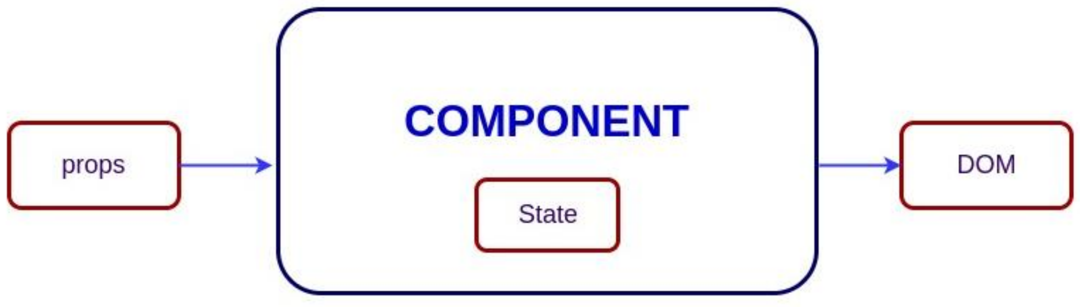
props are received from a parent component and are **read only**

Although the data in props is accessible to a component, React philosophy is that props should be immutable and top-down. What this means is that a parent component can pass on whatever data it wants to its children as props, but the child component cannot modify its props. So, if you try to edit the props as I did below, you will get the "Cannot assign to read-only" TypeError.

```
const ViewInterns = (interns) => {
    // props are read only
interns.map((intern) => (
intern.id=1001;//error
          ))}
…
}
```

State

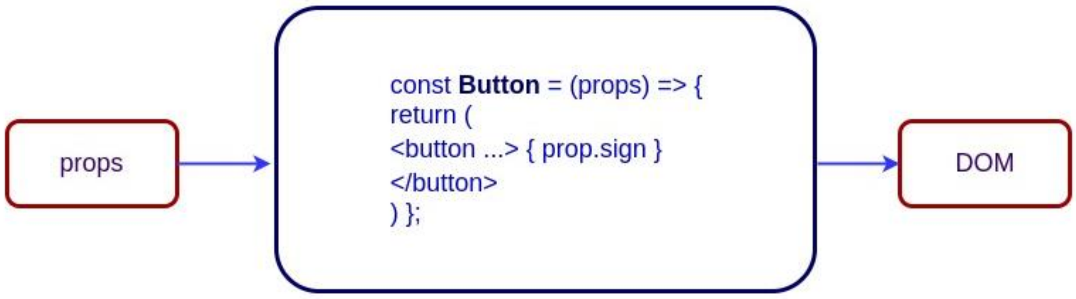
State, on the other hand, is an object that is owned by the component where it is declared. Its scope is limited to the current component. A component can initialize its state and update it whenever necessary. The state of the parent component usually ends up being props of the child component. When the state is passed out of the current scope, we refer to it as a prop.



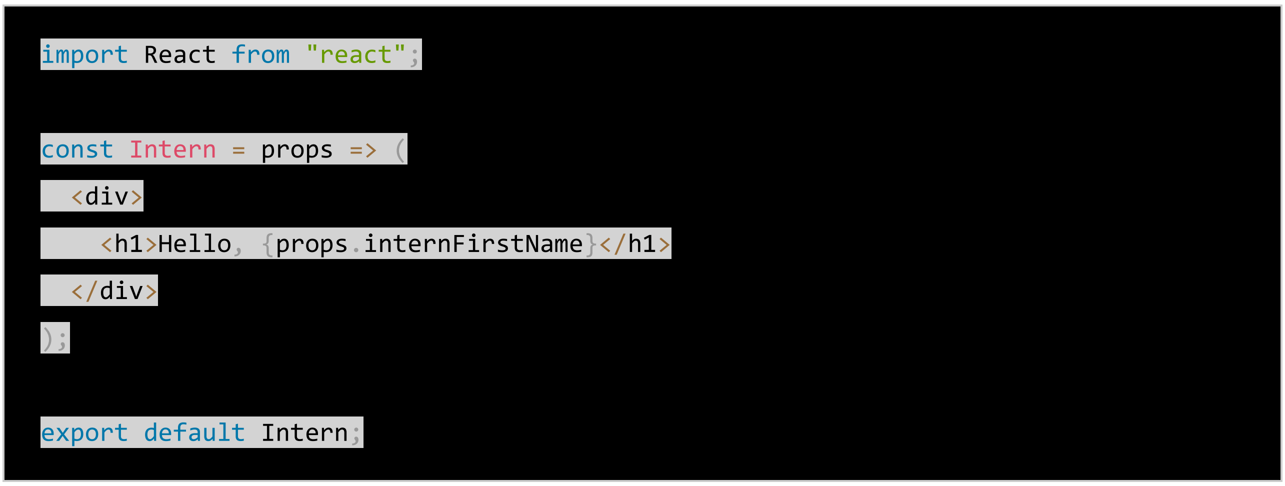
state is used for internal communication inside a Component

Functional Components

- Functional components are basic JavaScript functions. These are typically arrow functions but can also be created with the regular `function` keyword.
- Sometimes referred to as “dumb” or “stateless” components as they simply accept data and display them in some form; that is they are mainly responsible for rendering UI.
- React lifecycle methods (for example, `componentDidMount`) cannot be used in functional components.
- There is no render method used in functional components.
- These are mainly responsible for UI and are typically presentational only (For example, a Button component).
- Functional components can accept and use props.
- Functional components should be favored if you do not need to make use of React state.



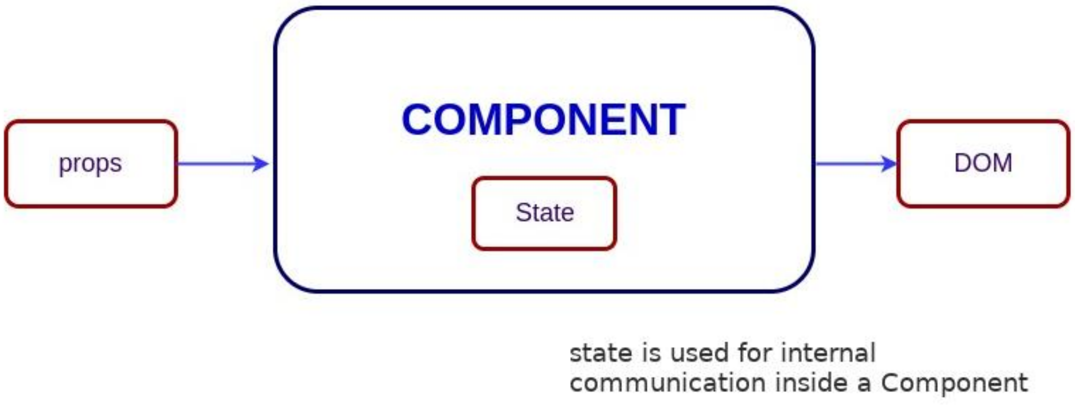
```
const Button = (props) => {
return (
<button ...> { prop.sign }
</button>
) };
```



```
import React from "react";

const Intern = props => (
  <div>
    <h1>Hello, {props.internFirstName}</h1>
  </div>
);

export default Intern;
```

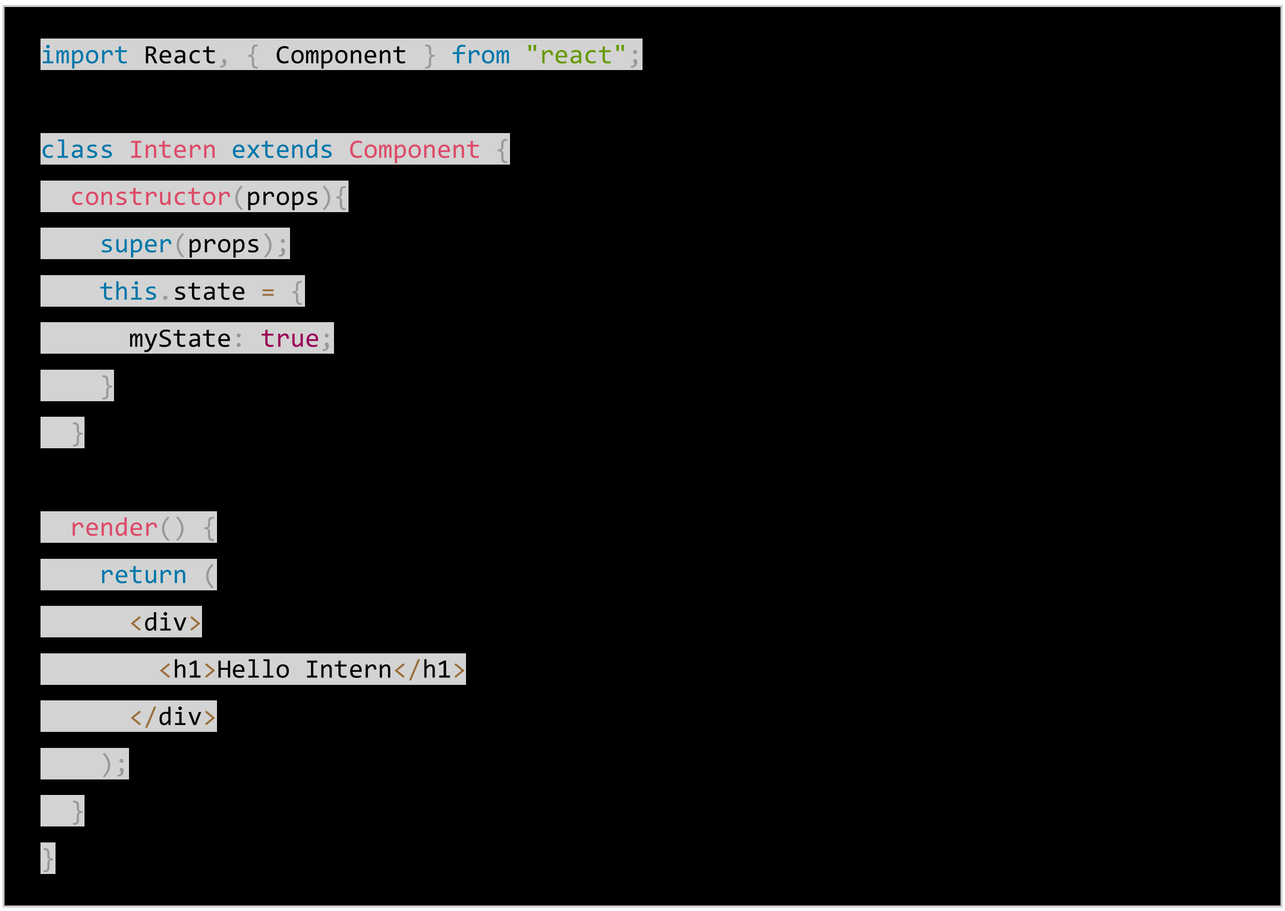
Class Components

- Class components make use of ES6 class and extend the Component class in React.
- Sometimes called “smart” or “stateful” components as they tend to implement logic and state.
- React lifecycle methods can be used inside class components (for example, componentDidMount).
- You pass props down to class components and access them with this.props



To define a React component class, you need to extend `React.Component`:

The only method you *must* define in a `React.Component` subclass is called `render()`



```
import React, { Component } from "react";

class Intern extends Component {
  constructor(props){
    super(props);
    this.state = {
      myState: true;
    }
  }

  render() {
    return (
      <div>
        <h1>Hello Intern</h1>
      </div>
    );
  }
}
```

```
export default Intern;
```

Class components should always call the base constructor with `props`.

Presentational Components

Presentational components are coupled with the view or how things look. These components accept props from their container counterpart and render them. Everything that has to do with describing the UI should go here.

Presentational components are reusable and should stay decoupled from the behavioral layer. A presentational component receives the data and callbacks exclusively via props and when an event occurs, like a button being pressed, it performs a callback to the container component via props to invoke an event handling method.

Container Components

Container components will deal with the behavioral part. A container component tells the presentational component what should be rendered using props. In container components. You should place your API calls and store the result into the component's state.

Pure Component

A component is said to be pure if it is guaranteed to return the same result given the same props and state. A functional component is a good example of a pure component because, given an input, you know what will be rendered.

```
const ViewInterns = ({intern}) => (
 <div>{` ${intern.id}`}</div>
);
```

Class components can be pure too as long as their props and state are immutable.

A React Component can go through four stages of its life as follows.

- **Initialization:** This is the stage where the component is constructed with the given Props and default state. This is done in the constructor of a Component Class

- **Mounting:** Mounting is the stage of rendering the JSX returned by the render method itself.

  componentWillMount()→render()-→componentDidMount()

- **Updating:** Updating is the stage when the state of a component is updated and the application is repainted.

  For props:

  componentWillRecieveProps()→shouldComponentUpdate()→(if true)→componentWillUpdate()→render()→componentDidUpdate()

  For state:

  setState()→shouldComponentUpdate()→(if true)→componentWillUpdate()→render()→componentDidUpdate()

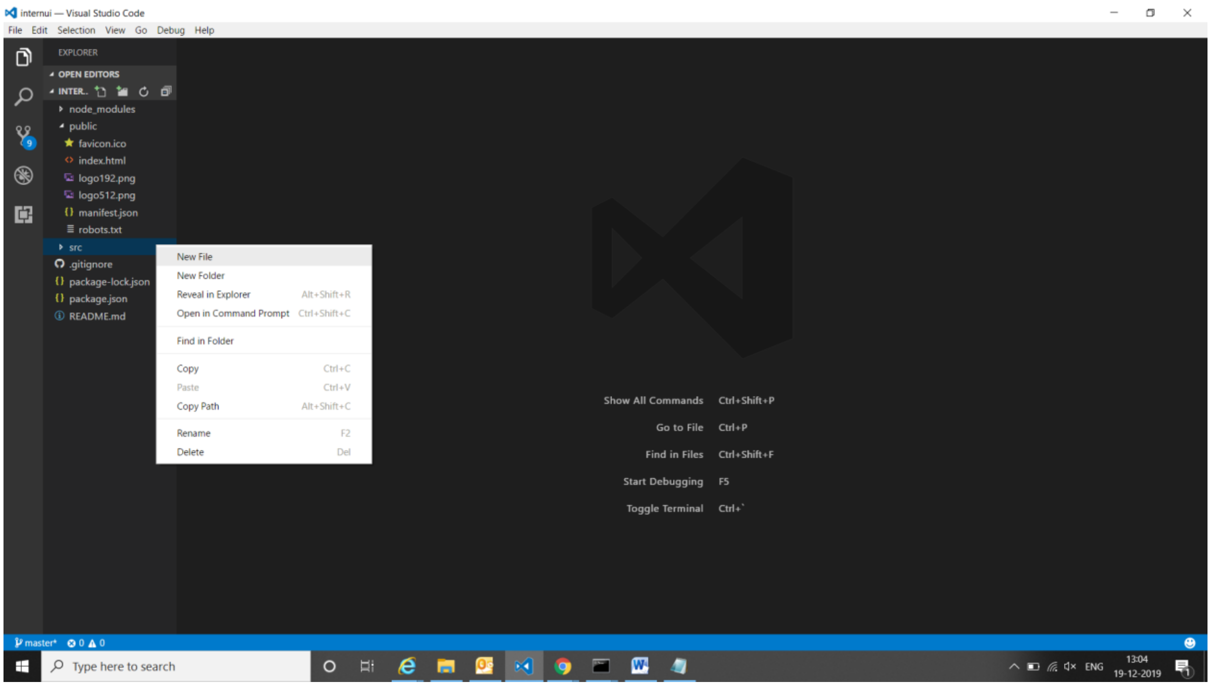
- **Unmounting:** As the name suggests Unmounting is the final step of the component lifecycle where the component is removed from the page.

  componentWillUnmount()

User-Defined Components Must Be Capitalized

When an element type starts with a lowercase letter, it refers to a built-in component like <div> or <span> and results in a string 'div' or 'span' passed to React.createElement. Types that start with a capital letter like <Foo /> compile to React.createElement(Foo) and correspond to a component defined or imported in your JavaScript file.

Create App.js in src folder



As an example to pass props from Root component to child component, we will retrieve all interns data from REST API in App component and pass it to ViewInterns component.

When root component will mount, we are calling REST API (SpringBootRest application InternBusinessApp)

We should ideally provide absolute URI instead of relative URI. We will provide relative URI in next section.

We retrieve all interns data and put it in state interns[] of App component.

```
componentDidMount() {
    console.log("component did mount");
  fetch("http://localhost:8089/interns-management/yash-interns")
  .then(res => res.json())
  .then((data) => {
          this.setState({ interns: data })
        })
         .catch(console.log);
}
```

To pass interns[] to ViewInterns component,

```
<Route path="/view">
            <ViewInterns interns={this.state.interns}/>
</Route>
```

React Router is the standard routing library for React. React Router keeps your UI in sync with the URL. It has a simple API with powerful features like lazy code loading, dynamic route matching, and location transition handling built right in.

Complete listing of App.js is as below,

```
import React from 'react';
import logo from './logo.svg';
import './App.css';
import {Component} from 'react';
import {ViewInterns} from './ViewInterns.js'
import {RegisterIntern} from './RegisterIntern.js'
import {UpdateInterns} from './UpdateInterns.js'
import {DeleteInterns} from './DeleteInterns.js'
import {UpdateInternsLevel} from './UpdateInternsLevel.js'
import {
  BrowserRouter as Router,
  Switch,
  Route,
  Link
} from "react-router-dom";

class App extends Component{
  state={

    interns:[]
  }

  componentDidMount() {
    console.log("component did mount");
  fetch("http://localhost:8089/interns-management/yash-interns")
  .then(res => res.json())
  .then((data) => {
          this.setState({ interns: data })
```

```
        })
         .catch(console.log);
}
 render() {
        return (
<Router>
      <div>
        <ul>
          <li>
            <Link to="/view">View Interns Details</Link>
          </li>
          <li>
            <Link to="/register">Register Interns Details</Link>
          </li>

           <li>
            <Link to="/update">Update Interns Details</Link>
          </li>
             <li>
            <Link to="/updatelevel">Update Interns Level</Link>
          </li>
           <li>
            <Link to="/delete">Delete Interns Details</Link>
          </li>
        </ul>
        <hr />
        <Switch>
          <Route path="/view">
            <ViewInterns interns={this.state.interns}/>
          </Route>
          <Route path="/register">
            <RegisterIntern />
          </Route>
          <Route path="/update">
            <UpdateInterns />
          </Route>
          <Route path="/updatelevel">
            <UpdateInternsLevel />
          </Route>
             <Route path="/delete">
            <DeleteInterns/>
          </Route>
        </Switch>
      </div>
    </Router>
     )
      }
}
```

```
export default App;
```

Create ViewInterns.js in src folder with below code,

```
import {Component} from 'react';
import React from 'react';

    const ViewInterns = ({ interns }) => {
      return (
        <div>
          <center><h1>Interns Information</h1></center>
          <table class="table table-striped">
                 <tr>
      <th>Intern Id</th>
      <th>Intern First Name</th>
      <th>Intern Last Name</th>
      <th>Intern Age</th>
      <th>Intern Level</th>

    </tr>
          {interns.map((intern) => (
               <tr>
      <td>{intern.id}</td>
      <td>{intern.internFirstName}</td>
      <td>{intern.internLastName}</td>
      <td>{intern.internAge}</td>
      <td>{intern.level}</td>
    </tr>
          ))}
          </table>
        </div>
      )
    };

    export default ViewInterns;
    export {ViewInterns}
```

To register intern data create a file RegisterIntern.js in src folder with below content,

```
import React, { Component } from 'react';

var panelStyle = {
  'max-width': '80%',
  margin: '0 auto'
}

class RegisterIntern extends Component {
    constructor(props) {
    super(props);
        this.handleSubmit = this.handleSubmit.bind(this);

    this.state = {
            formFields:{id:
'',internFirstName:'',internLastName:'',internAge:'',semester1Marks:'',semeste
r2Marks:'',semester3Marks:''},
            res:''
    };
  }

handleSubmit(event) {
    event.preventDefault();
    fetch('http://localhost:8089/interns-management/yash-interns', {
      method: 'POST',
      body: JSON.stringify({
        id: this.state.formFields.id,
        internFirstName:this.state.formFields.internFirstName,
        internLastName:this.state.formFields.internLastName,
        internAge:this.state.formFields.internAge,
        semester1Marks:this.state.formFields.semester1Marks,
        semester2Marks:this.state.formFields.semester2Marks,
        semester3Marks:this.state.formFields.semester3Marks
      })
        ,
        headers: {
        "Content-type": "application/json; charset=UTF-8"
      }
    }).then(this.state.res='Registration of Intern successful');
  }


 inputChangeHandler = (event) => {
   let formFields = {...this.state.formFields};
   formFields[event.target.name] = event.target.value;
   this.setState({
    formFields
   });
 }


  render() {
    return(
     <div>
    <div class="panel panel-primary" style={panelStyle}>
      <div class="panel panel-heading">Intern Registration</div>
      <div class="panel panel-body">
        <form onSubmit={this.handleSubmit}>
          <strong>Id:</strong> <br /> <input type="text" name="id"
placeholder="Id" onChange={(e) => this.inputChangeHandler.call(this, e)}
value={this.state.formFields.id} /> <br />
          <strong>First Name:</strong> <br /> <input type="text"
name="internFirstName" placeholder="First Name" onChange={(e) =>
this.inputChangeHandler.call(this, e)}
value={this.state.formFields.internFirstName}/> <br />
          <strong>Last Name:</strong> <br /> <input type="text"
name="internLastName" placeholder="Last Name" onChange={(e) =>
this.inputChangeHandler.call(this, e)}
value={this.state.formFields.internLastName}/> <br />
          <strong>Age:</strong> <br /> <input type="text" name="internAge"
placeholder="age" onChange={(e) => this.inputChangeHandler.call(this, e)}
value={this.state.formFields.internAge}/> <br />
          <strong>Semester1:</strong> <br /> <input type="text"
name="semester1Marks" placeholder="marks1" onChange={(e) =>
this.inputChangeHandler.call(this, e)}
value={this.state.formFields.semester1Marks} /> <br /><br />
          <strong>Semester2:</strong> <br /> <input type="text"
name="semester2Marks" placeholder="marks2"onChange={(e) =>
this.inputChangeHandler.call(this, e)}
value={this.state.formFields.semester2Marks} /> <br /><br />
          <strong>Semester3:</strong> <br /> <input type="text"
name="semester3Marks" placeholder="marks3" onChange={(e) =>
this.inputChangeHandler.call(this, e)}
value={this.state.formFields.semester3Marks}/> <br /><br />
 <button class="btn btn-primary">Register Intern</button>
 </form>
  <p> {this.state.res}</p>

      </div>
    </div>
  </div>

    );
  }
}
export default RegisterIntern
export {RegisterIntern}
```

We are specifying intial state formFields in constructor with default values.

```
    this.state = {
            formFields:{id:
'',internFirstName:'',internLastName:'',internAge:'',semester1Marks:'',semeste
r2Marks:'',semester3Marks:''},
            res:''
    };
```

On value change event on input box, we call function inputChangeHandler,

```
inputChangeHandler = (event) => {
   let formFields = {...this.state.formFields};
   formFields[event.target.name] = event.target.value;
   this.setState({
    formFields
   });
 }
```

…this.state.formFields is spread syntax.

The spread syntax allows an expression to be expanded in places where multiple arguments (for function calls) or multiple elements (for array literals) or multiple variables (for destructuring assignment) are expected.

Variables declared with the let keyword can have Block Scope.

We are assigning each formFields element with a value(user's input) and changing the state of formFields using this.setState

```
   this.setState({
    formFields
   });
```

On click on submitting form we are calling **handleSubmit()** and posting the request to Rest API using fetch() with HTTP Method POST

```
handleSubmit(event) {
    event.preventDefault();
    fetch('http://localhost:8089/interns-management/yash-interns', {
      method: 'POST',
      body: JSON.stringify({
        id: this.state.formFields.id,
        internFirstName:this.state.formFields.internFirstName,
        internLastName:this.state.formFields.internLastName,
        internAge:this.state.formFields.internAge,
        semester1Marks:this.state.formFields.semester1Marks,
        semester2Marks:this.state.formFields.semester2Marks,
        semester3Marks:this.state.formFields.semester3Marks
      })
        ,
        headers: {
        "Content-type": "application/json; charset=UTF-8"
      }
    }).then(this.state.res='Registration of Intern successful')
         .catch(console.log);
    this.forceUpdate();
  }
```

Fetch is a promise-based API which returns a response object. So, we make use of the json() method to get the response object which is stored in data and used to update the state of users in our application

To update intern data create UpdateInterns.js in src folder

```
import React, { Component } from 'react';
var panelStyle = {
    'max-width': '80%',
    margin: '0 auto'
}

class UpdateInterns extends Component {
    constructor(props) {
        super(props);
        this.handleSubmit = this.handleSubmit.bind(this);

        this.state = {
            formFields:{id:
'',internFirstName:'',internLastName:'',internAge:'',semester1Marks:'',semeste
r2Marks:'',semester3Marks:''},
            res:''
    };

    }


handleSubmit(event) {
  console.log("handle submit");
    event.preventDefault();
    fetch('http://localhost:8089/interns-management/yash-interns-manage', {
      method: 'PUT',
      body: JSON.stringify({
                id: this.state.formFields.id,
        internFirstName:this.state.formFields.internFirstName,
        internLastName:this.state.formFields.internLastName,
        internAge:this.state.formFields.internAge,
        semester1Marks:this.state.formFields.semester1Marks,
        semester2Marks:this.state.formFields.semester2Marks,
        semester3Marks:this.state.formFields.semester3Marks
      })
        ,
        headers: {
                "Content-type": "application/json; charset=UTF-8"
            }
    }).then(this.state.res='Updation of Intern successful')


  }


 inputChangeHandler = (event) => {
   let formFields = {...this.state.formFields};
   formFields[event.target.name] = event.target.value;
   this.setState({
    formFields
   });
 }



    render() {
    return(
     <div>
    <div class="panel panel-primary" style={panelStyle}>
      <div class="panel panel-heading">Intern Update</div>
      <div class="panel panel-body">
        <form onSubmit={this.handleSubmit}>
          <strong>Id:</strong> <br /> <input type="text" name="id"
placeholder="Id" onChange={(e) => this.inputChangeHandler.call(this, e)}
value={this.state.formFields.id} /> <br />
          <strong>First Name:</strong> <br /> <input type="text"
name="internFirstName" placeholder="First Name" onChange={(e) =>
this.inputChangeHandler.call(this, e)}
value={this.state.formFields.internFirstName}/> <br />
          <strong>Last Name:</strong> <br /> <input type="text"
name="internLastName" placeholder="Last Name" onChange={(e) =>
this.inputChangeHandler.call(this, e)}
value={this.state.formFields.internLastName}/> <br />
          <strong>Age:</strong> <br /> <input type="text" name="internAge"
placeholder="age" onChange={(e) => this.inputChangeHandler.call(this, e)}
value={this.state.formFields.internAge}/> <br />
          <strong>Semester1:</strong> <br /> <input type="text"
name="semester1Marks" placeholder="marks1" onChange={(e) =>
this.inputChangeHandler.call(this, e)}
value={this.state.formFields.semester1Marks} /> <br /><br />
          <strong>Semester2:</strong> <br /> <input type="text"
name="semester2Marks" placeholder="marks2"onChange={(e) =>
this.inputChangeHandler.call(this, e)}
value={this.state.formFields.semester2Marks} /> <br /><br />
          <strong>Semester3:</strong> <br /> <input type="text"
name="semester3Marks" placeholder="marks3" onChange={(e) =>
this.inputChangeHandler.call(this, e)}
value={this.state.formFields.semester3Marks}/> <br /><br />
 <button class="btn btn-primary">Update Intern</button>
 </form>

 <p>{this.state.res}</p>
      </div>
    </div>
  </div>
```

```
    );
  }
}

export default UpdateInterns
export {UpdateInterns}
```

To update interns Level, create a file UpdateInternsLevel.js in src folder,

```
import React, { Component } from 'react';
var panelStyle = {
    'max-width': '80%',
    margin: '0 auto'
}

class UpdateInternsLevel extends Component {
    constructor(props) {
        super(props);
        this.handleSubmit = this.handleSubmit.bind(this);

        this.state = {
            formFields:{id:
'',internFirstName:'',internLastName:'',internAge:'',semester1Marks:'',semeste
r2Marks:'',semester3Marks:''},
            res:''
    };

    }


handleSubmit(event) {
  console.log("handle submit");
    event.preventDefault();
    fetch('http://localhost:8089/interns-management/yash-interns-level', {
      method: 'PATCH',
      body: JSON.stringify({
              id: this.state.formFields.id,
        semester1Marks:this.state.formFields.semester1Marks,
        semester2Marks:this.state.formFields.semester2Marks,
        semester3Marks:this.state.formFields.semester3Marks
      })
        ,
        headers: {
              "Content-type": "application/json; charset=UTF-8"
            }
    }).then(this.state.res='Updation of Intern level successful');
  }


 inputChangeHandler = (event) => {
   let formFields = {...this.state.formFields};
   formFields[event.target.name] = event.target.value;
   this.setState({
    formFields
   });
 }



    render() {
    return(
     <div>
    <div class="panel panel-primary" style={panelStyle}>
      <div class="panel panel-heading">Update Intern  Level</div>
      <div class="panel panel-body">
        <form onSubmit={this.handleSubmit}>
          <strong>Id:</strong> <br /> <input type="text" name="id"
placeholder="Id" onChange={(e) => this.inputChangeHandler.call(this, e)}
value={this.state.formFields.id} /> <br />
           <strong>Semester1:</strong> <br /> <input type="text"
name="semester1Marks" placeholder="marks1" onChange={(e) =>
this.inputChangeHandler.call(this, e)}
value={this.state.formFields.semester1Marks} /> <br /><br />
          <strong>Semester2:</strong> <br /> <input type="text"
name="semester2Marks" placeholder="marks2"onChange={(e) =>
this.inputChangeHandler.call(this, e)}
value={this.state.formFields.semester2Marks} /> <br /><br />
          <strong>Semester3:</strong> <br /> <input type="text"
name="semester3Marks" placeholder="marks3" onChange={(e) =>
this.inputChangeHandler.call(this, e)}
value={this.state.formFields.semester3Marks}/> <br /><br />
 <button class="btn btn-primary">Update Intern Level</button>
 </form>
  <p> {this.state.res}</p>

      </div>
    </div>
  </div>

    );
  }
}

export default UpdateInternsLevel
export {UpdateInternsLevel}
```

To delete interns data create DeleteInterns.js in src folder,

```
import React, { Component } from 'react';
var panelStyle = {
  'max-width': '80%',
  margin: '0 auto'
}

class DeleteInterns extends Component {
    constructor(props) {
    super(props);
        this.handleSubmit = this.handleSubmit.bind(this);

    this.state = {
            formFields:{id: ''},
            res:''
    };

  }

    handleSubmit(event) {
  console.log("handle submit");
    event.preventDefault();
    fetch('http://localhost:8089/interns-management/yash-interns-
manage/'+this.state.formFields.id, {
      method: 'DELETE'
      }).then(this.state.res='Deletion of Intern successful')

  }


 inputChangeHandler = (event) => {
   let formFields = {...this.state.formFields};
   formFields[event.target.name] = event.target.value;
   this.setState({
    formFields
   });
 }


  render() {
    return(
     <div>
    <div class="panel panel-primary" style={panelStyle}>
      <div class="panel panel-heading">Intern Delete</div>
      <div class="panel panel-body">
        <form onSubmit={this.handleSubmit}>
          <strong>Id:</strong> <br /> <input type="text" name="id"
placeholder="Id" onChange={(e) => this.inputChangeHandler.call(this, e)}
value={this.state.formFields.id} /> <br />

 <button class="btn btn-primary">Delete Intern</button>
 </form>
  <p> {this.state.res}</p>

      </div>
    </div>
  </div>

    );
  }
}


export default DeleteInterns
export {DeleteInterns}
```

Before we test our end to end application, Make sure your Spring REST boot or JAX-RS application is running and able to connect with Mongo DB.

I will test our UI using Spring REST Boot business application with MongoDB created in earlier section.

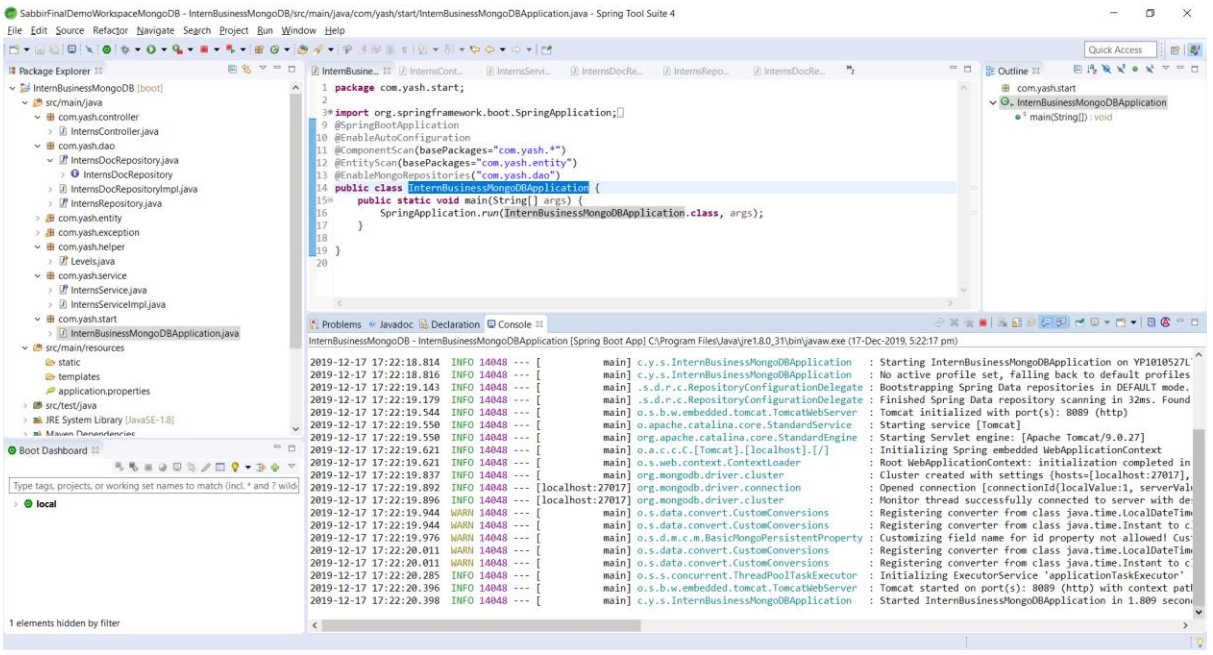
Enabling CORS

This **@CrossOrigin** annotation enables cross-origin requests only for this specific method. By default, its allows all origins, all headers, the HTTP methods specified in the **@RequestMapping** annotation and a maxAge of 30 minutes is used. You can customize this behaviour by specifying the value of one of the annotation attributes: origins, methods, allowedHeaders, exposedHeaders, allowCredentials or max Age

In InternsController class apply below annotation,

```
@CrossOrigin(origins = "http://localhost:3000")
public class InternsController {
```

Start InternBusinessApplication by running com.yash.start.
`InternBusinessMongoDBApplication`



SabbirFinalDemoWorkspaceMongoDB - InternBusinessMongoDB/src/main/java/com/yash/start/InternBusinessMongoDBApplication.java - Spring Tool Suite 4

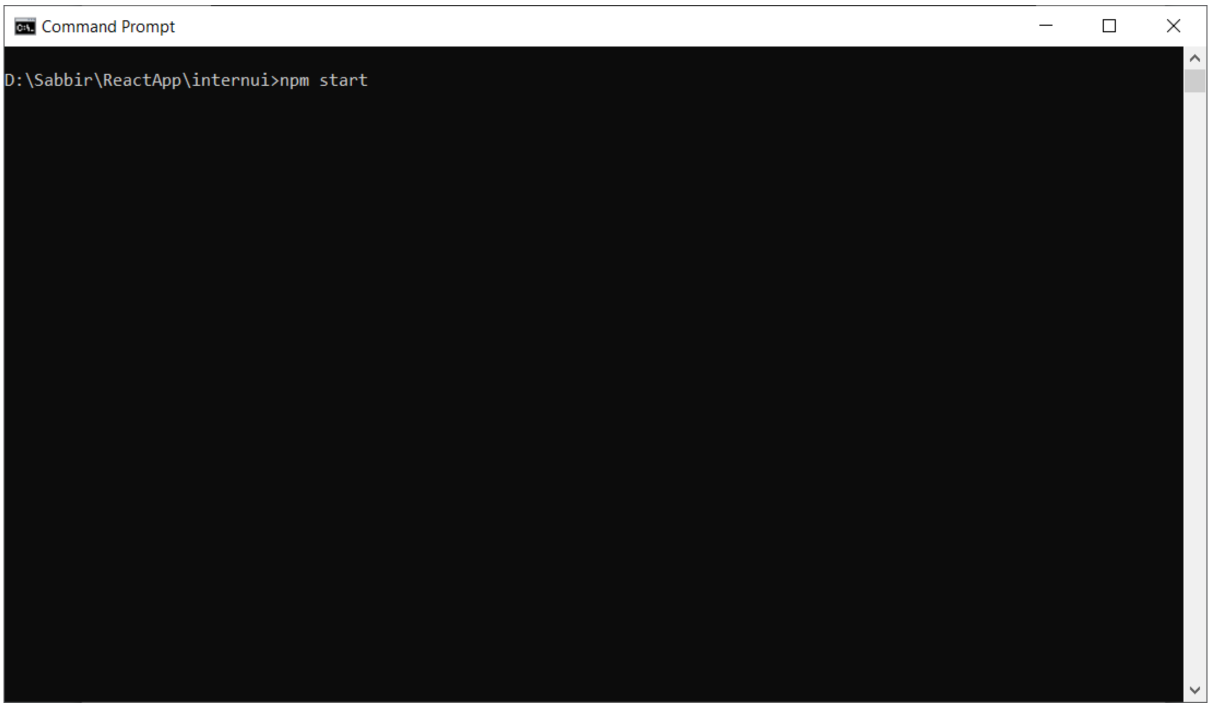
```java
package com.yash.start;

import org.springframework.boot.SpringApplication;
@SpringBootApplication
@EnableAutoConfiguration
@ComponentScan(basePackages="com.yash.*")
@EntityScan(basePackages="com.yash.entity")
@EnableMongoRepositories("com.yash.dao")
public class InternBusinessMongoDBApplication {
    public static void main(String[] args) {
        SpringApplication.run(InternBusinessMongoDBApplication.class, args);
    }

}
```

InternBusinessMongoDB - InternBusinessMongoDBApplication [Spring Boot App] C:\Program Files\Java\jre1.8.0_31\bin\javaw.exe (17-Dec-2019, 5:22:17 pm)

```
2019-12-17 17:22:18.814  INFO 14048 --- [           main] c.y.s.InternBusinessMongoDBApplication   : Starting InternBusinessMongoDBApplication on YP1010527L
2019-12-17 17:22:18.816  INFO 14048 --- [           main] c.y.s.InternBusinessMongoDBApplication   : No active profile set, falling back to default profiles
2019-12-17 17:22:19.143  INFO 14048 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data repositories in DEFAULT mode.
2019-12-17 17:22:19.179  INFO 14048 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 32ms. Found
2019-12-17 17:22:19.544  INFO 14048 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8089 (http)
2019-12-17 17:22:19.550  INFO 14048 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2019-12-17 17:22:19.550  INFO 14048 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.27]
2019-12-17 17:22:19.621  INFO 14048 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2019-12-17 17:22:19.621  INFO 14048 --- [           main] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed in
2019-12-17 17:22:19.837  INFO 14048 --- [           main] org.mongodb.driver.cluster               : Cluster created with settings {hosts=[localhost:27017],
2019-12-17 17:22:19.892  INFO 14048 --- [localhost:27017] org.mongodb.driver.connection            : Opened connection [connectionId{localValue:1, serverVal
2019-12-17 17:22:19.896  INFO 14048 --- [localhost:27017] org.mongodb.driver.cluster               : Monitor thread successfully connected to server with de
2019-12-17 17:22:19.944  WARN 14048 --- [           main] o.s.data.convert.CustomConversions       : Registering converter from class java.time.LocalDateTim
2019-12-17 17:22:19.944  WARN 14048 --- [           main] o.s.data.convert.CustomConversions       : Registering converter from class java.time.Instant to c
2019-12-17 17:22:19.976  WARN 14048 --- [           main] o.s.d.m.c.m.BasicMongoPersistentProperty : Customizing field name for id property not allowed! Cus
2019-12-17 17:22:20.011  WARN 14048 --- [           main] o.s.data.convert.CustomConversions       : Registering converter from class java.time.LocalDateTim
2019-12-17 17:22:20.011  WARN 14048 --- [           main] o.s.data.convert.CustomConversions       : Registering converter from class java.time.Instant to c
2019-12-17 17:22:20.285  INFO 14048 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
2019-12-17 17:22:20.396  INFO 14048 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8089 (http) with context pat
2019-12-17 17:22:20.398  INFO 14048 --- [           main] c.y.s.InternBusinessMongoDBApplication   : Started InternBusinessMongoDBApplication in 1.809 secon
```

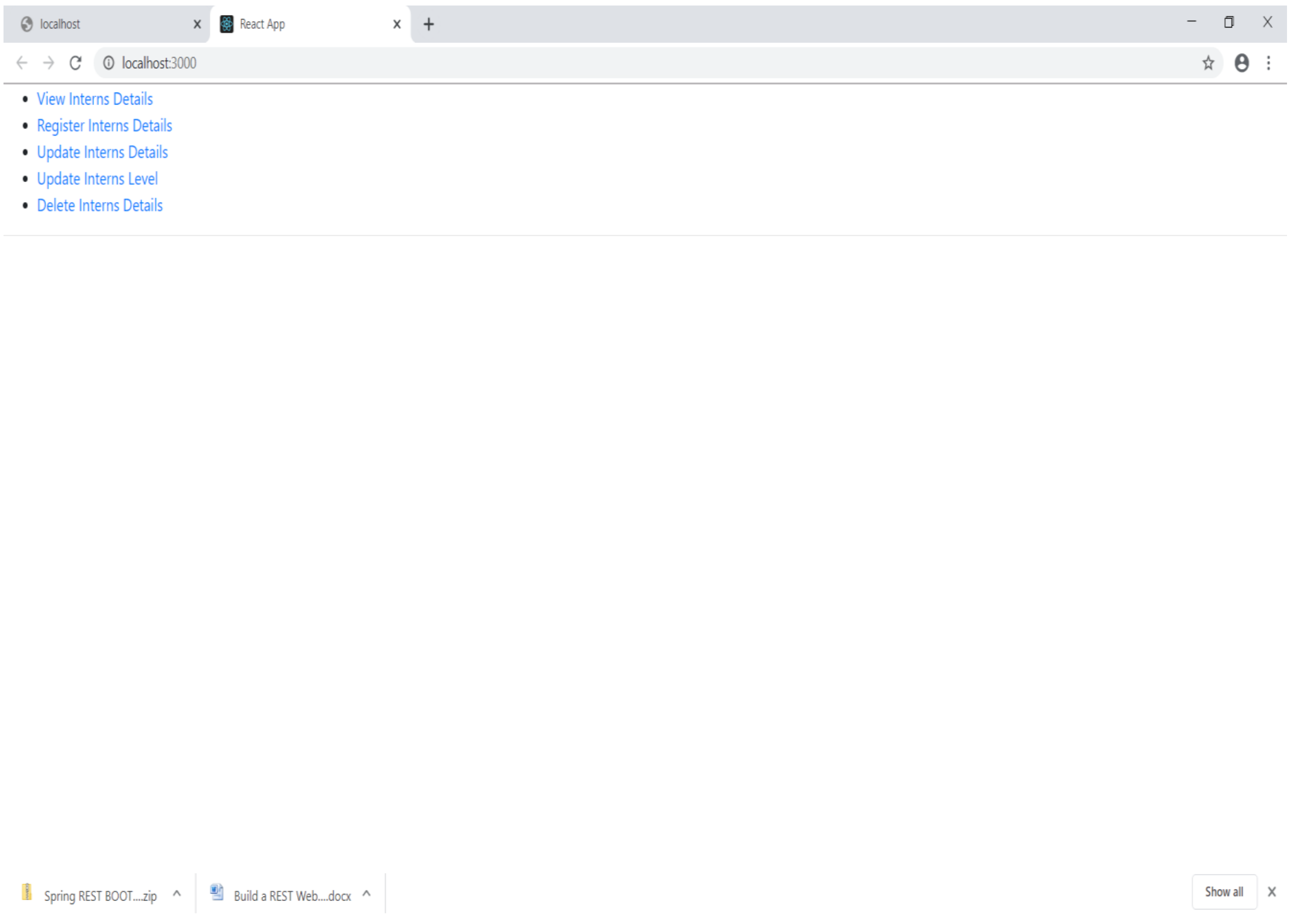
Run react application,

Type: npm start



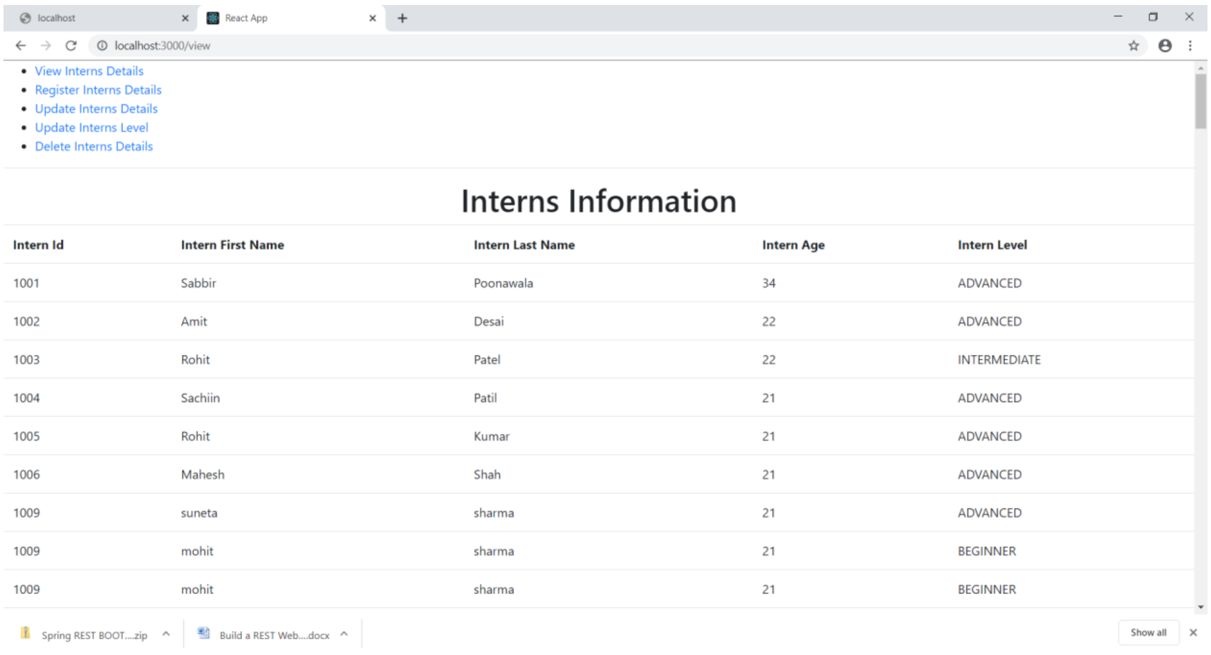
Command Prompt

```
D:\Sabbir\ReactApp\internui>npm start
```



- View Interns Details
- Register Interns Details
- Update Interns Details
- Update Interns Level
- Delete Interns Details

Click on view interns to view all interns data,



localhost | React App

localhost:3000/view

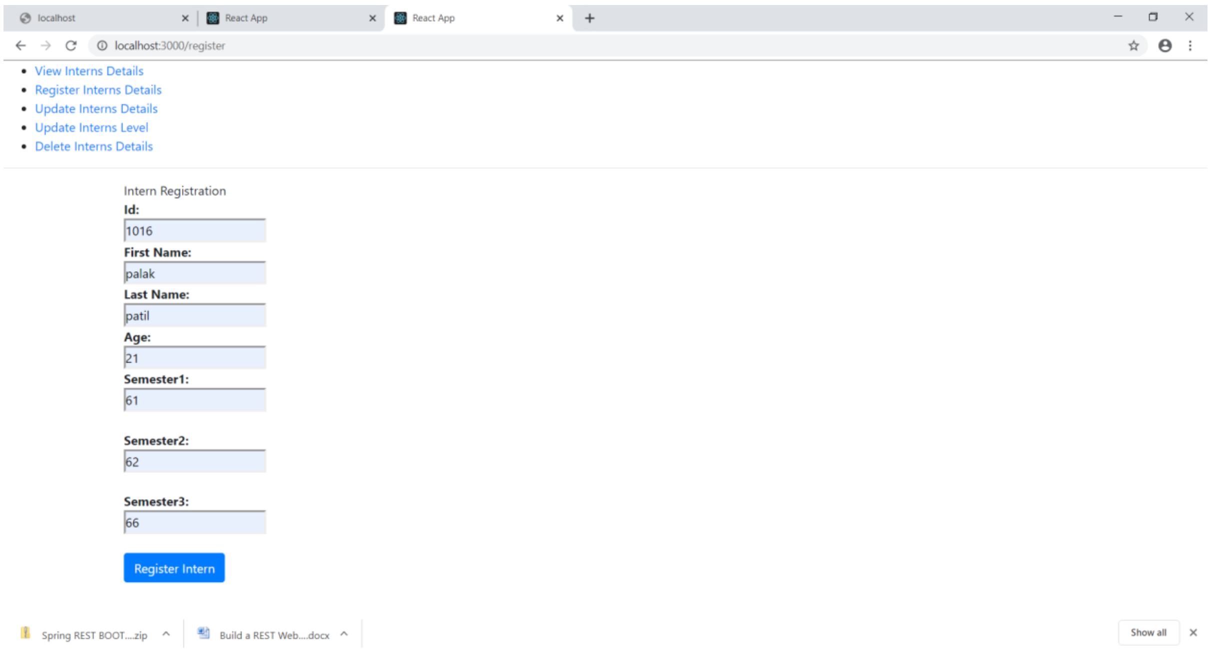
- View Interns Details
- Register Interns Details
- Update Interns Details
- Update Interns Level
- Delete Interns Details

Interns Information

Intern Id	Intern First Name	Intern Last Name	Intern Age	Intern Level
1001	Sabbir	Poonawala	34	ADVANCED
1002	Amit	Desai	22	ADVANCED
1003	Rohit	Patel	22	INTERMEDIATE
1004	Sachiin	Patil	21	ADVANCED
1005	Rohit	Kumar	21	ADVANCED
1006	Mahesh	Shah	21	ADVANCED
1009	suneta	sharma	21	ADVANCED
1009	mohit	sharma	21	BEGINNER
1009	mohit	sharma	21	BEGINNER

Spring REST BOOT....zip | Build a REST Web....docx | Show all

To Register intern details click on link Register Intern details,



localhost | React App | React App

localhost:3000/register

- View Interns Details
- Register Interns Details
- Update Interns Details
- Update Interns Level
- Delete Interns Details

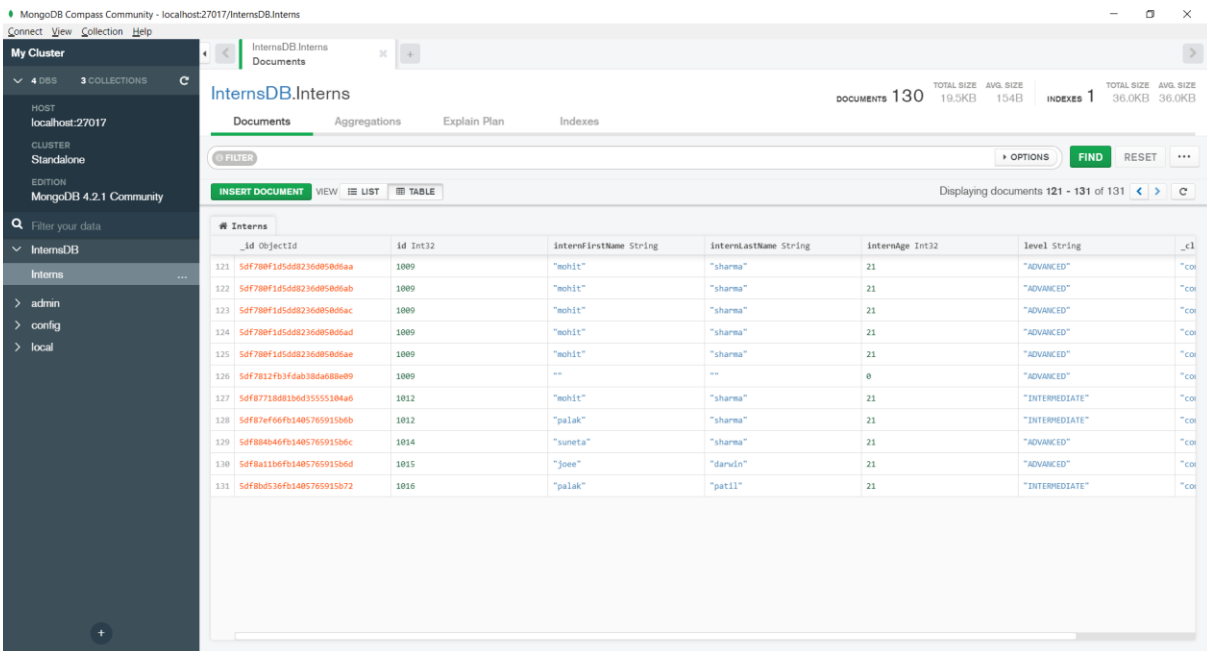
Intern Registration

Id:	1016
First Name:	palak
Last Name:	patil
Age:	21
Semester1:	61
Semester2:	62
Semester3:	66

Register Intern

Spring REST BOOT....zip | Build a REST Web....docx | Show all

Verify if record is inserted in mongo db,


MongoDB Compass Community - localhost:27017/InternsDB.Interns

InternsDB.Interns — DOCUMENTS 130, TOTAL SIZE 19.5KB, AVG. SIZE 154B, INDEXES 1, TOTAL SIZE 36.0KB, AVG. SIZE 36.0KB

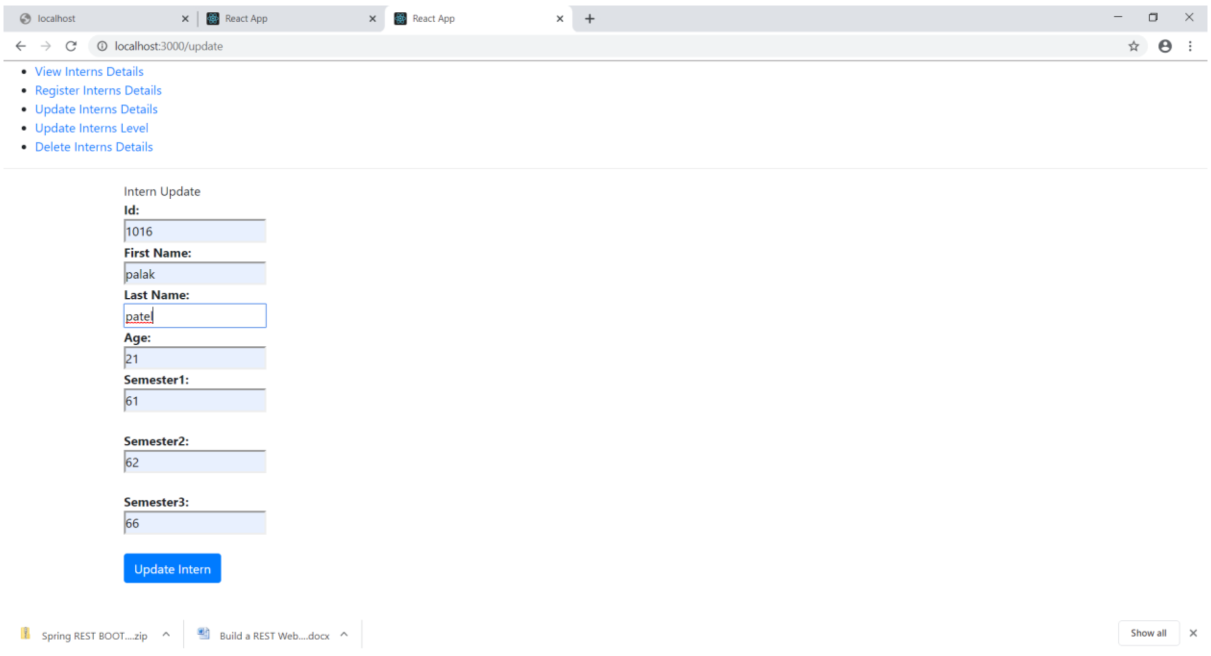
HOST localhost:27017 · CLUSTER Standalone · EDITION MongoDB 4.2.1 Community

Displaying documents 121 - 131 of 131

	_id ObjectId	id Int32	internFirstName String	internLastName String	internAge Int32	level String	_cl
121	5df780f1d5dd8236d050d6aa	1009	"mohit"	"sharma"	21	"ADVANCED"	"co
122	5df780f1d5dd8236d050d6ab	1009	"mohit"	"sharma"	21	"ADVANCED"	"co
123	5df780f1d5dd8236d050d6ac	1009	"mohit"	"sharma"	21	"ADVANCED"	"co
124	5df780f1d5dd8236d050d6ad	1009	"mohit"	"sharma"	21	"ADVANCED"	"co
125	5df780f1d5dd8236d050d6ae	1009	"mohit"	"sharma"	21	"ADVANCED"	"co
126	5df7812fb3fdab38da688e09	1009	""	""	0	"ADVANCED"	"co
127	5df87718d81b6d35555104a6	1012	"mohit"	"sharma"	21	"INTERMEDIATE"	"co
128	5df87ef66fb1405765915b6b	1012	"palak"	"sharma"	21	"INTERMEDIATE"	"co
129	5df884b46fb1405765915b6c	1014	"suneta"	"sharma"	21	"ADVANCED"	"co
130	5df8a11b6fb1405765915b6d	1015	"joee"	"darwin"	21	"ADVANCED"	"co
131	5df8bd536fb1405765915b72	1016	"palak"	"patil"	21	"INTERMEDIATE"	"co

Enter details,

To Update Intern details click on link Update Intern details,


localhost:3000/update

- View Interns Details
- Register Interns Details
- Update Interns Details
- Update Interns Level
- Delete Interns Details

Intern Update

Id: 1016

First Name: palak

Last Name: patel

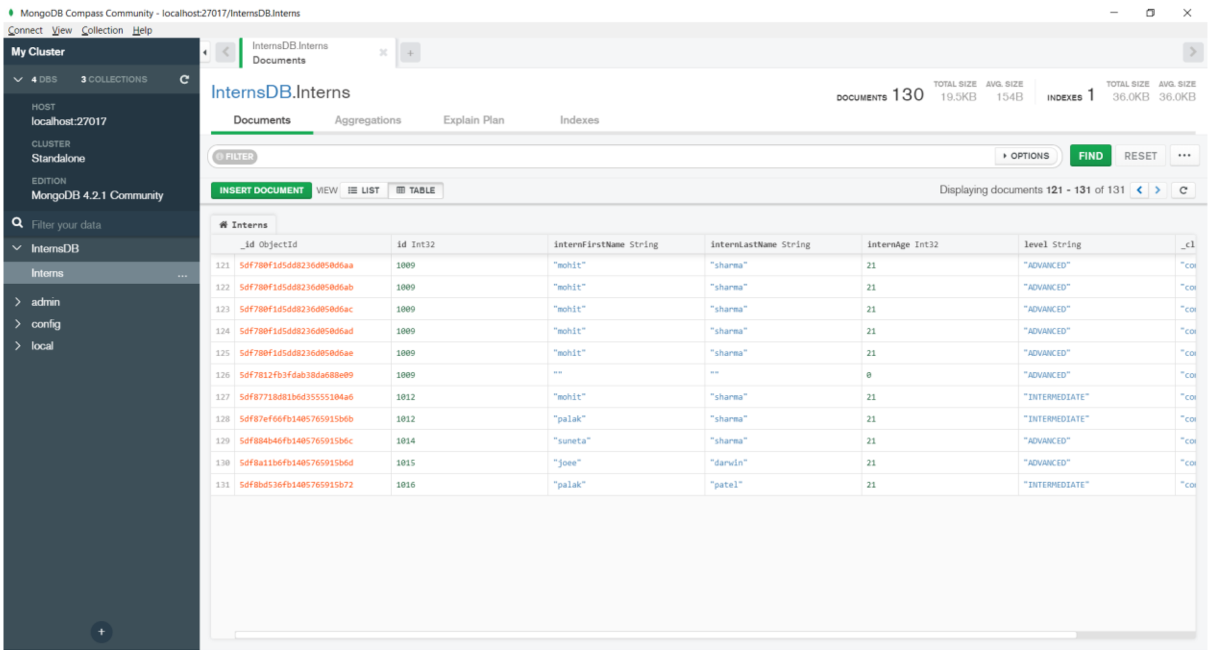
Age: 21

Semester1: 61

Semester2: 62

Semester3: 66

Update Intern


MongoDB Compass Community - localhost:27017/InternsDB.Interns

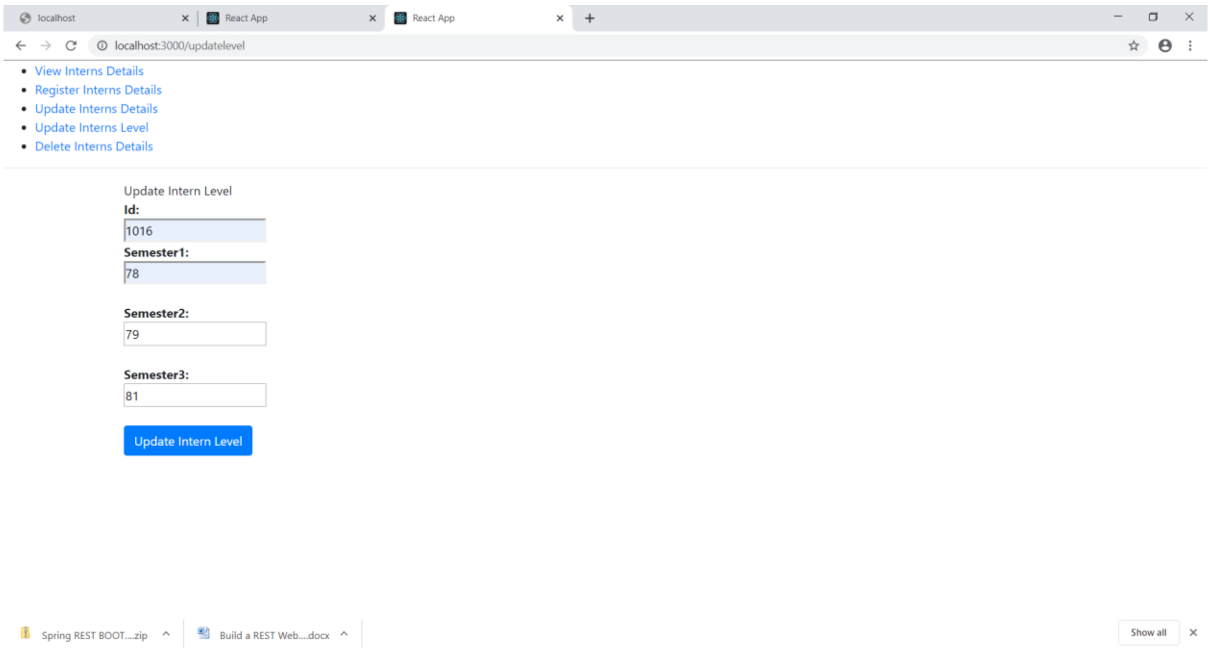
InternsDB.Interns

DOCUMENTS 130 TOTAL SIZE 19.5KB AVG. SIZE 154B INDEXES 1 TOTAL SIZE 36.0KB AVG. SIZE 36.0KB

Displaying documents 121 - 131 of 131

	_id ObjectId	id Int32	internFirstName String	internLastName String	internAge Int32	level String	_cl
121	5df780f1d5dd8236d050d6aa	1009	"mohit"	"sharma"	21	"ADVANCED"	"co
122	5df780f1d5dd8236d050d6ab	1009	"mohit"	"sharma"	21	"ADVANCED"	"co
123	5df780f1d5dd8236d050d6ac	1009	"mohit"	"sharma"	21	"ADVANCED"	"co
124	5df780f1d5dd8236d050d6ad	1009	"mohit"	"sharma"	21	"ADVANCED"	"co
125	5df780f1d5dd8236d050d6ae	1009	"mohit"	"sharma"	21	"ADVANCED"	"co
126	5df7812fb3fdab38da688e09	1009	""	""	0	"ADVANCED"	"co
127	5df87718d81b6d35555104a6	1012	"mohit"	"sharma"	21	"INTERMEDIATE"	"co
128	5df87ef66fb1405765915b6b	1012	"palak"	"sharma"	21	"INTERMEDIATE"	"co
129	5df884b46fb1405765915b6c	1014	"suneta"	"sharma"	21	"ADVANCED"	"co
130	5df8a11b6fb1405765915b6d	1015	"joee"	"darwin"	21	"ADVANCED"	"co
131	5df8bd536fb1405765915b72	1016	"palak"	"patel"	21	"INTERMEDIATE"	"co

To update intern's level, click on link Update Intern Level,


localhost:3000/updatelevel

- View Interns Details
- Register Interns Details
- Update Interns Details
- Update Interns Level
- Delete Interns Details

Update Intern Level

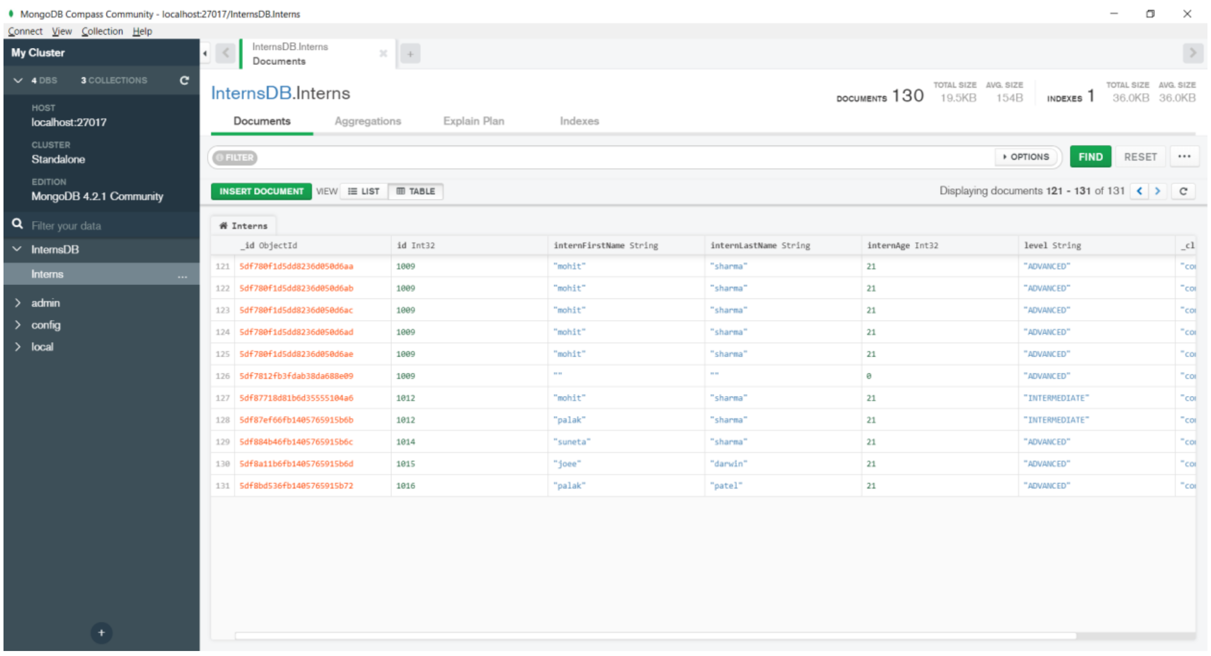
Id: 1016

Semester1: 78

Semester2: 79

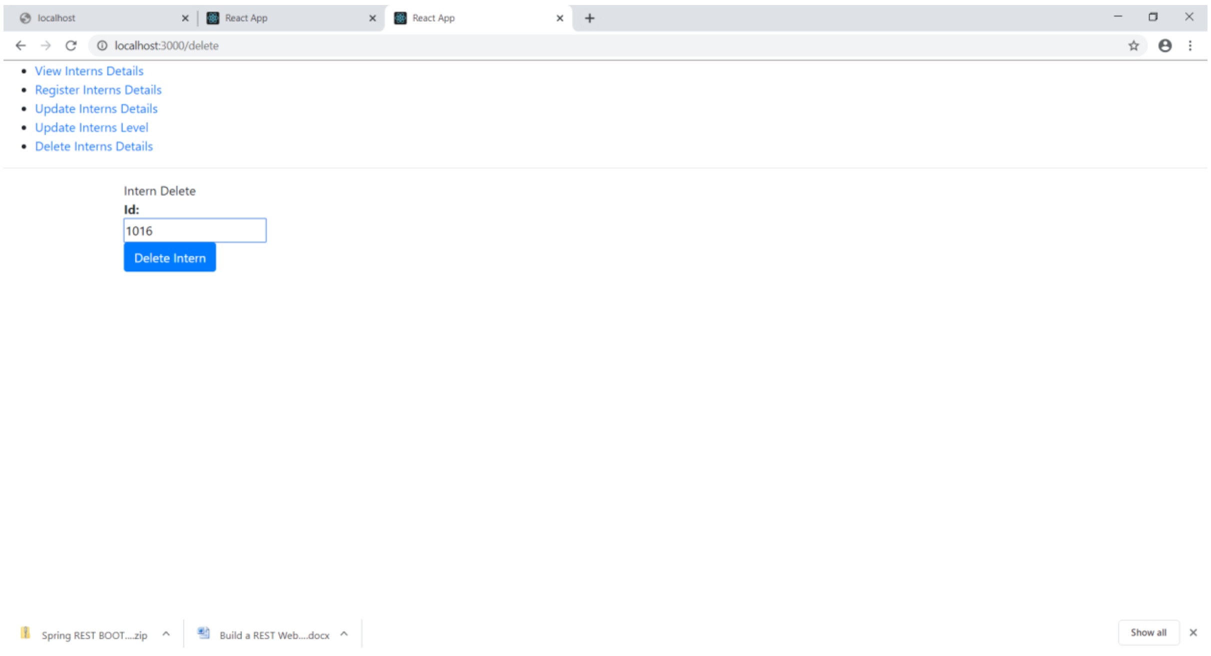
Semester3: 81

Update Intern Level



	_id ObjectId	id Int32	internFirstName String	internLastName String	internAge Int32	level String	_cl
121	5df780f1d5dd8236d050d6aa	1009	"mohit"	"sharma"	21	"ADVANCED"	"co
122	5df780f1d5dd8236d050d6ab	1009	"mohit"	"sharma"	21	"ADVANCED"	"co
123	5df780f1d5dd8236d050d6ac	1009	"mohit"	"sharma"	21	"ADVANCED"	"co
124	5df780f1d5dd8236d050d6ad	1009	"mohit"	"sharma"	21	"ADVANCED"	"co
125	5df780f1d5dd8236d050d6ae	1009	"mohit"	"sharma"	21	"ADVANCED"	"co
126	5df7812fb3fdab38da688e09	1009	""	""	0	"ADVANCED"	"co
127	5df87718d81b6d35555104a6	1012	"mohit"	"sharma"	21	"INTERMEDIATE"	"co
128	5df87ef66fb1405765915b6b	1012	"palak"	"sharma"	21	"INTERMEDIATE"	"co
129	5df884b46fb1405765915b6c	1014	"suneta"	"sharma"	21	"ADVANCED"	"co
130	5df8a11b6fb1405765915b6d	1015	"joee"	"darwin"	21	"ADVANCED"	"co
131	5df8bd536fb1405765915b72	1016	"palak"	"patel"	21	"ADVANCED"	"co

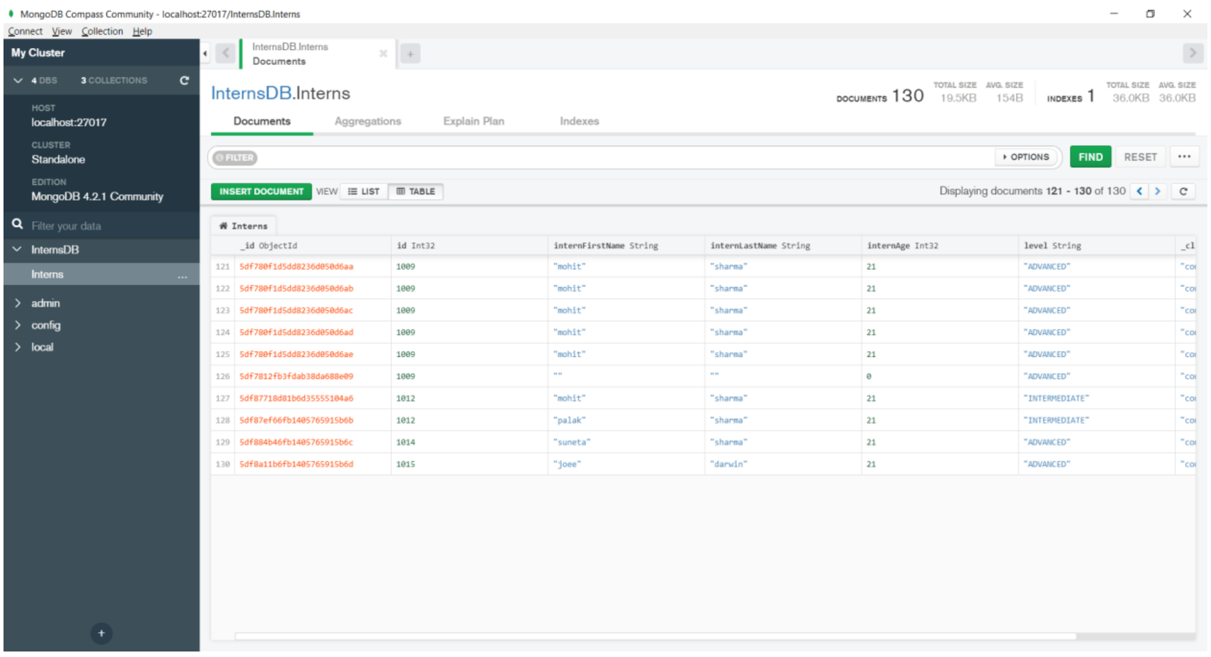
To delete intern's data click on link delete intern details and enter intern id,



- View Interns Details
- Register Interns Details
- Update Interns Details
- Update Interns Level
- Delete Interns Details

Intern Delete
Id:
1016
Delete Intern

Verify in MongoDB if record is deleted.


MongoDB Compass Community - localhost:27017/InternsDB.Interns
Connect View Collection Help
My Cluster
4 DBS 3 COLLECTIONS
HOST localhost:27017
CLUSTER Standalone
EDITION MongoDB 4.2.1 Community
Filter your data
InternsDB
Interns
admin
config
local
InternsDB.Interns Documents
InternsDB.Interns
DOCUMENTS 130 TOTAL SIZE 19.5KB AVG. SIZE 154B
INDEXES 1 TOTAL SIZE 36.0KB AVG. SIZE 36.0KB
Documents
Aggregations
Explain Plan
Indexes
FILTER
OPTIONS
FIND
RESET
INSERT DOCUMENT
VIEW
LIST
TABLE
Displaying documents 121 - 130 of 130
Interns

	_id ObjectId	id Int32	internFirstName String	internLastName String	internAge Int32	level String	_cl
121	5df780f1d5dd8236d050d6aa	1009	"mohit"	"sharma"	21	"ADVANCED"	"co
122	5df780f1d5dd8236d050d6ab	1009	"mohit"	"sharma"	21	"ADVANCED"	"co
123	5df780f1d5dd8236d050d6ac	1009	"mohit"	"sharma"	21	"ADVANCED"	"co
124	5df780f1d5dd8236d050d6ad	1009	"mohit"	"sharma"	21	"ADVANCED"	"co
125	5df780f1d5dd8236d050d6ae	1009	"mohit"	"sharma"	21	"ADVANCED"	"co
126	5df7812fb3fdab38da688e09	1009	""	""	0	"ADVANCED"	"co
127	5df87718d81b6d35555104a6	1012	"mohit"	"sharma"	21	"INTERMEDIATE"	"co
128	5df87ef66fb1405765915b6b	1012	"palak"	"sharma"	21	"INTERMEDIATE"	"co
129	5df884b46fb1405765915b6c	1014	"suneta"	"sharma"	21	"ADVANCED"	"co
130	5df8a11b6fb1405765915b6d	1015	"joee"	"darwin"	21	"ADVANCED"	"co

React Form validation

Inputs given by end-user in user interface have to be thoroughly validated. After user interface validation, data will be sent to business application where business rules will be applied.

We will apply validation in RegisterIntern.js, UpdateInterns.js, UpdateInternLevel.js and DeleteInterns.js

We will apply below validations,

Id: must be at least of 4 digits
First Name: cannot be blank
Last Name: cannot be blank
Age: must be greater than 18
Semester marks: must be positive or zero

Validation on user input could also be based on data in data stores.

For example while registering intern if inter id is also ready assigned it should display error “Intern Id already exists”. We will perform this in next section.

Specifiy state errors in component with default values,

```
errors:{
              id:'',
              internFirstName:'',
              internLastName:'',
              internAge:'',
              semester1Marks:'',
              semester2Marks:'',
              semester3Marks:''
            },
```

and a flag for enabling or disabling button,

```
buttonEnabled:true
```

When input in textbox will change, validate input based on validation criteria's using switch,

```
switch (name) {
       case 'id':
         errors.id =
        value.length < 4
          ? 'Id  must be 4 digits long!'
          : '';
          break;
        case 'internFirstName':
        errors.internFirstName=value.length<=0 ?'First Name cannot be
blank':'';
       break;
       case 'internLastName':
        errors.internLastName=value.length<=0 ?'Last Name cannot be blank':'';
        break;
        case 'internAge':
        errors.internAge=value<=18  ?'Age cannot be less than 18':'';
        break;
         case 'semester1Marks':
        errors.semester1Marks=value<0  ?'Semester 1 marks cannot be
negative':'';
        break;
          case 'semester2Marks':
        errors.semester2Marks=value<0  ?'Semester 2 marks cannot be
negative':'';
        break;
          case 'semester3Marks':
        errors.semester3Marks=value<0  ?'Semester 3 marks cannot be
negative':'';
        break;

     }
```

If validation fails, state errors' fields will change with a value,

errors.id="id must be 4 digits long!"

if any field in errors has a value i.e value.length >0 return true;

```
const validateForm = (errors) => {
  let valid = true;
  Object.values(errors).forEach(
    // if we have an error string set valid to false
    (val) => val.length > 0 && (valid = false)
  );
  return valid;
}
```

If validateForm return true, change state buttonEnabled as false

```
  if(validateForm(this.state.errors)) {
       console.info('Valid Form')
       this.state.buttonEnabled=false;

  }else{
       console.error('Invalid Form')
```

To display validation error message,

```
<span style={spanStyle}>{this.state.errors.id}</span><br />
```

spanStyle is for styling span,

```
var spanStyle={
  color:'red'
}
```

Complete listing of RegisterIntern.js

```
import React, { Component } from 'react';
var panelStyle = {
  'max-width': '80%',
  margin: '0 auto',
}
var spanStyle={
  color:'red'
}
```

```
class RegisterIntern extends Component {
    constructor(props) {
    super(props);
        this.handleSubmit = this.handleSubmit.bind(this);
    this.state = {
            formFields:{id:
'',internFirstName:'',internLastName:'',internAge:'',semester1Marks:'',semeste
r2Marks:'',semester3Marks:''},
            res:'' ,
            errors:{
              id:'',
              internFirstName:'',
              internLastName:'',
              internAge:'',
              semester1Marks:'',
              semester2Marks:'',
              semester3Marks:''
            },
            buttonEnabled:true
    };
  }
handleSubmit(event) {
    event.preventDefault();
    fetch('http://localhost:8089/interns-management/yash-interns', {
      method: 'POST',
      body: JSON.stringify({
        id: this.state.formFields.id,
        internFirstName:this.state.formFields.internFirstName,
        internLastName:this.state.formFields.internLastName,
        internAge:this.state.formFields.internAge,
        semester1Marks:this.state.formFields.semester1Marks,
        semester2Marks:this.state.formFields.semester2Marks,
        semester3Marks:this.state.formFields.semester3Marks
      })
        ,
        headers: {
        "Content-type": "application/json; charset=UTF-8"
      }
    }).then(this.state.res='Registration of Intern successful')
         .catch(console.log);
    this.forceUpdate();
  }
 inputChangeHandler = (event) => {
   let formFields = {...this.state.formFields};
   formFields[event.target.name] = event.target.value;
   this.setState({
    formFields
   });

  const validateForm = (errors) => {
  let valid = true;
  Object.values(errors).forEach(
    // if we have an error string set valid to false
    (val) => val.length > 0 && (valid = false)
  );
  return valid;
}

const { name, value } = event.target;
     let errors = this.state.errors;
     this.setState({errors, [name]: value}, ()=> {
      console.log(errors)
  })

switch (name) {
       case 'id':
         errors.id =
        value.length < 4
          ? 'Id  must be 4 digits long!'
          : '';
          break;
        case 'internFirstName':
        errors.internFirstName=value.length<=0 ?'First Name cannot be
blank':'';
       break;
       case 'internLastName':
        errors.internLastName=value.length<=0 ?'Last Name cannot be blank':'';
        break;
        case 'internAge':
        errors.internAge=value<=18  ?'Age cannot be less than 18':'';
        break;
         case 'semester1Marks':
        errors.semester1Marks=value<0  ?'Semester 1 marks cannot be
negative':'';
        break;
          case 'semester2Marks':
        errors.semester2Marks=value<0  ?'Semester 2 marks cannot be
negative':'';
        break;
          case 'semester3Marks':
        errors.semester3Marks=value<0  ?'Semester 3 marks cannot be
negative':'';
        break;

     }
```

```
      if(validateForm(this.state.errors)) {
       console.info('Valid Form')
       this.state.buttonEnabled=false;

  }else{
       console.error('Invalid Form')
       this.state.buttonEnabled=true;
  }
 }
  render() {
    return(
     <div>
    <div class="panel panel-primary" style={panelStyle}>
      <div class="panel panel-heading">Intern Registration</div>
      <div class="panel panel-body">
        <form onSubmit={this.handleSubmit}>
          <strong>Id:</strong> <br /> <input type="text" name="id"
placeholder="Id" onChange={(e) => this.inputChangeHandler.call(this, e)}
value={this.state.formFields.id} />&nbsp;<span
style={spanStyle}>{this.state.errors.id}</span><br />
          <strong>First Name:</strong> <br /> <input type="text"
name="internFirstName" onChange={(e) => this.inputChangeHandler.call(this, e)}
value={this.state.formFields.internFirstName}/>&nbsp;<span
style={spanStyle}>{this.state.errors.internFirstName}</span> <br />
          <strong>Last Name:</strong> <br /> <input type="text"
name="internLastName" placeholder="Last Name" onChange={(e) =>
this.inputChangeHandler.call(this, e)}
value={this.state.formFields.internLastName}/> &nbsp;<span
style={spanStyle}>{this.state.errors.internLastName}</span><br />
          <strong>Age:</strong> <br /> <input type="text" name="internAge"
placeholder="age" onChange={(e) => this.inputChangeHandler.call(this, e)}
value={this.state.formFields.internAge}/>&nbsp;<span
style={spanStyle}>{this.state.errors.internAge}</span> <br />
          <strong>Semester1:</strong> <br /> <input type="text"
name="semester1Marks" placeholder="marks1" onChange={(e) =>
this.inputChangeHandler.call(this, e)}
value={this.state.formFields.semester1Marks} />&nbsp;<span
style={spanStyle}>{this.state.errors.semester1Marks}</span> <br /><br />
          <strong>Semester2:</strong> <br /> <input type="text"
name="semester2Marks" placeholder="marks2"onChange={(e) =>
this.inputChangeHandler.call(this, e)}
value={this.state.formFields.semester2Marks} />&nbsp;<span
style={spanStyle}>{this.state.errors.semester2Marks}</span> <br /><br />
          <strong>Semester3:</strong> <br /> <input type="text"
name="semester3Marks" placeholder="marks3" onChange={(e) =>
this.inputChangeHandler.call(this, e)}
value={this.state.formFields.semester3Marks}/>&nbsp;<span
style={spanStyle}>{this.state.errors.semester3Marks}</span> <br /><br />
 <button class="btn btn-primary" disabled={this.state.buttonEnabled}>Register
Intern</button>
 </form>
  <p> {this.state.res}</p>
      </div>
    </div>
  </div>
    );
  }
}
export default RegisterIntern
export {RegisterIntern}
```

Complete listing of UpdateInterns.js

```
import React, { Component } from 'react';
var panelStyle = {
  'max-width': '80%',
  margin: '0 auto'
}

var spanStyle={
  color:'red'
}

class UpdateInterns extends Component {
    constructor(props) {
    super(props);
        this.handleSubmit = this.handleSubmit.bind(this);

    this.state = {
            formFields:{id:
'',internFirstName:'',internLastName:'',internAge:'',semester1Marks:'',semeste
r2Marks:'',semester3Marks:''},
            res:'' ,
              errors:{
              id:'',
              internFirstName:'',
              internLastName:'',
              internAge:'',
              semester1Marks:'',
              semester2Marks:'',
              semester3Marks:''
            },
            buttonEnabled:true
    };

  }


handleSubmit(event) {
  console.log("handle submit");
    event.preventDefault();
    fetch('http://localhost:8089/interns-management/yash-interns-manage', {
      method: 'PUT',
      body: JSON.stringify({
        id: this.state.formFields.id,
        internFirstName:this.state.formFields.internFirstName,
        internLastName:this.state.formFields.internLastName,
        internAge:this.state.formFields.internAge,
        semester1Marks:this.state.formFields.semester1Marks,
        semester2Marks:this.state.formFields.semester2Marks,
        semester3Marks:this.state.formFields.semester3Marks
      })
        ,
        headers: {
        "Content-type": "application/json; charset=UTF-8"
      }
    }).then(this.state.res='Updation of Intern
successful').catch(console.log())
this.forceUpdate();

  }


 inputChangeHandler = (event) => {
   let formFields = {...this.state.formFields};
   formFields[event.target.name] = event.target.value;
   this.setState({
    formFields
   });

  const validateForm = (errors) => {
  let valid = true;
  Object.values(errors).forEach(
    // if we have an error string set valid to false
    (val) => val.length > 0 && (valid = false)
  );
  return valid;
}

const { name, value } = event.target;
     let errors = this.state.errors;
     this.setState({errors, [name]: value}, ()=> {
      console.log(errors)
  })

switch (name) {
       case 'id':
         errors.id =
        value.length < 4
          ? 'Id  must be 4 digits long!'
          : '';
          break;
        case 'internFirstName':
        errors.internFirstName=value.length<=0 ?'First Name cannot be
blank':'';
       break;
       case 'internLastName':
        errors.internLastName=value.length<=0 ?'Last Name cannot be blank':'';
        break;
        case 'internAge':
        errors.internAge=value<=18  ?'Age cannot be less than 18':'';
        break;
         case 'semester1Marks':
        errors.semester1Marks=value<0  ?'Semester 1 marks cannot be
negative':'';
        break;
          case 'semester2Marks':
        errors.semester2Marks=value<0  ?'Semester 2 marks cannot be
negative':'';
        break;
          case 'semester3Marks':
        errors.semester3Marks=value<0  ?'Semester 3 marks cannot be
negative':'';
        break;

     }

      if(validateForm(this.state.errors)) {
       console.info('Valid Form')
       this.state.buttonEnabled=false;

  }else{
       console.error('Invalid Form')
       this.state.buttonEnabled=true;
  }
 }



    render() {
```

```
    return(
     <div>
    <div class="panel panel-primary" style={panelStyle}>
      <div class="panel panel-heading">Intern Update</div>
      <div class="panel panel-body">
       <form onSubmit={this.handleSubmit}>
          <strong>Id:</strong> <br /> <input type="text" name="id"
placeholder="Id" onChange={(e) => this.inputChangeHandler.call(this, e)}
value={this.state.formFields.id} />&nbsp;<span
style={spanStyle}>{this.state.errors.id}</span><br />
          <strong>First Name:</strong> <br /> <input type="text"
name="internFirstName" onChange={(e) => this.inputChangeHandler.call(this, e)}
value={this.state.formFields.internFirstName}/>&nbsp;<span
style={spanStyle}>{this.state.errors.internFirstName}</span> <br />
          <strong>Last Name:</strong> <br /> <input type="text"
name="internLastName" placeholder="Last Name" onChange={(e) =>
this.inputChangeHandler.call(this, e)}
value={this.state.formFields.internLastName}/> &nbsp;<span
style={spanStyle}>{this.state.errors.internLastName}</span><br />
          <strong>Age:</strong> <br /> <input type="text" name="internAge"
placeholder="age" onChange={(e) => this.inputChangeHandler.call(this, e)}
value={this.state.formFields.internAge}/>&nbsp;<span
style={spanStyle}>{this.state.errors.internAge}</span> <br />
          <strong>Semester1:</strong> <br /> <input type="text"
name="semester1Marks" placeholder="marks1" onChange={(e) =>
this.inputChangeHandler.call(this, e)}
value={this.state.formFields.semester1Marks} />&nbsp;<span
style={spanStyle}>{this.state.errors.semester1Marks}</span> <br /><br />
          <strong>Semester2:</strong> <br /> <input type="text"
name="semester2Marks" placeholder="marks2"onChange={(e) =>
this.inputChangeHandler.call(this, e)}
value={this.state.formFields.semester2Marks} />&nbsp;<span
style={spanStyle}>{this.state.errors.semester2Marks}</span> <br /><br />
          <strong>Semester3:</strong> <br /> <input type="text"
name="semester3Marks" placeholder="marks3" onChange={(e) =>
this.inputChangeHandler.call(this, e)}
value={this.state.formFields.semester3Marks}/>&nbsp;<span
style={spanStyle}>{this.state.errors.semester3Marks}</span> <br /><br />
 <button class="btn btn-primary" disabled={this.state.buttonEnabled}>Update
Intern</button>
 </form>

 <p>{this.state.res}</p>
      </div>
    </div>
  </div>

    );
  }
}

export default UpdateInterns
export {UpdateInterns}
```

Complete listing of UpdateInternsLevel.js,

```
import React, { Component } from 'react';
var panelStyle = {
  'max-width': '80%',
  margin: '0 auto'
}
var spanStyle={
  color:'red'
}
class UpdateInternsLevel extends Component {
    constructor(props) {
    super(props);
        this.handleSubmit = this.handleSubmit.bind(this);
        this.state = {
            formFields:{id:
'',semester1Marks:'',semester2Marks:'',semester3Marks:''},
            res:'',
             errors:{
              id:'',
              semester1Marks:'',
              semester2Marks:'',
              semester3Marks:''
            },
            buttonEnabled:true
    };

  }
handleSubmit(event) {
  console.log("handle submit");
    event.preventDefault();
    fetch('http://localhost:8089/interns-management/yash-interns-level', {
      method: 'PATCH',
      body: JSON.stringify({
        id: this.state.formFields.id,
        semester1Marks:this.state.formFields.semester1Marks,
        semester2Marks:this.state.formFields.semester2Marks,
        semester3Marks:this.state.formFields.semester3Marks
      })
        ,
        headers: {
        "Content-type": "application/json; charset=UTF-8"
      }
    }).then(this.state.res='Updation of Intern level successful')
    .catch(console.log);
        this.forceUpdate();
  }
 inputChangeHandler = (event) => {
   let formFields = {...this.state.formFields};
   formFields[event.target.name] = event.target.value;
   this.setState({
    formFields
   });
   const validateForm = (errors) => {
  let valid = true;
  Object.values(errors).forEach(
    // if we have an error string set valid to false
    (val) => val.length > 0 && (valid = false)
  );
  return valid;
}

const { name, value } = event.target;
     let errors = this.state.errors;
     this.setState({errors, [name]: value}, ()=> {
      console.log(errors)
  })

switch (name) {
       case 'id':
         errors.id =
        value.length < 4
          ? 'Id  must be 4 digits long!'
          : '';
          break;
        case 'internFirstName':
        errors.internFirstName=value.length<=0 ?'First Name cannot be 
blank':'';
       break;
       case 'internLastName':
        errors.internLastName=value.length<=0 ?'Last Name cannot be blank':'';
        break;
        case 'internAge':
        errors.internAge=value<=18  ?'Age cannot be less than 18':'';
        break;
         case 'semester1Marks':
        errors.semester1Marks=value<0  ?'Semester 1 marks cannot be 
negative':'';
```

```
        break;
          case 'semester2Marks':
        errors.semester2Marks=value<0  ?'Semester 2 marks cannot be
negative':'';
        break;
          case 'semester3Marks':
        errors.semester3Marks=value<0  ?'Semester 3 marks cannot be
negative':'';
        break;
     }
      if(validateForm(this.state.errors)) {
       console.info('Valid Form')
       this.state.buttonEnabled=false;

  }else{
       console.error('Invalid Form')
       this.state.buttonEnabled=true;
  }
 }
    render() {
    return(
     <div>
    <div class="panel panel-primary" style={panelStyle}>
      <div class="panel panel-heading">Update Intern  Level</div>
      <div class="panel panel-body">
        <form onSubmit={this.handleSubmit}>
        <strong>Id:</strong> <br /> <input type="text" name="id"
placeholder="Id" onChange={(e) => this.inputChangeHandler.call(this, e)}
value={this.state.formFields.id} />&nbsp;<span
style={spanStyle}>{this.state.errors.id}</span><br />
          <strong>Semester1:</strong> <br /> <input type="text"
name="semester1Marks" placeholder="marks1" onChange={(e) =>
this.inputChangeHandler.call(this, e)}
value={this.state.formFields.semester1Marks} />&nbsp;<span
style={spanStyle}>{this.state.errors.semester1Marks}</span> <br /><br />
          <strong>Semester2:</strong> <br /> <input type="text"
name="semester2Marks" placeholder="marks2"onChange={(e) =>
this.inputChangeHandler.call(this, e)}
value={this.state.formFields.semester2Marks} />&nbsp;<span style={spanStyle}
>{this.state.errors.semester2Marks}</span> <br /><br />
          <strong>Semester3:</strong> <br /> <input type="text"
name="semester3Marks" placeholder="marks3" onChange={(e) =>
this.inputChangeHandler.call(this, e)}
value={this.state.formFields.semester3Marks}/>&nbsp;<span
style={spanStyle}>{this.state.errors.semester3Marks}</span> <br /><br />
 <button class="btn btn-primary" disabled={this.state.buttonEnabled}>Update
Intern Level</button>
 </form>
```

```
  <p> {this.state.res}</p>

      </div>
    </div>
  </div>

    );
  }
}

export default UpdateInternsLevel
export {UpdateInternsLevel}
```

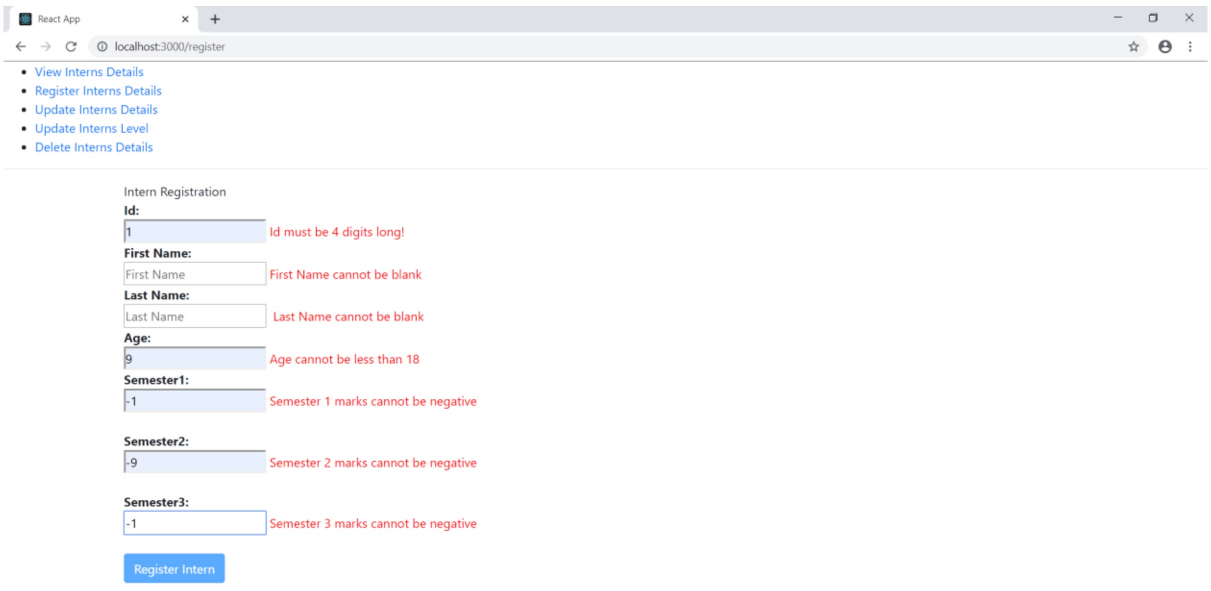
Complete listing of DeleteInterns.js,

```
import React, { Component } from 'react';
var panelStyle = {
  'max-width': '80%',
  margin: '0 auto'
}
var spanStyle={
  color:'red'
}

class DeleteInterns extends Component {
    constructor(props) {
    super(props);
        this.handleSubmit = this.handleSubmit.bind(this);

    this.state = {
            formFields:{id: ''},
            res:'',
              errors:{
              id:''
            },
            buttonEnabled:true
    };

  }
    handleSubmit(event) {
  console.log("handle submit");
    event.preventDefault();
    fetch('http://localhost:8089/interns-management/yash-interns-
manage/'+this.state.formFields.id, {
      method: 'DELETE'
    }).then(this.state.res='Deletion of Intern
successful').catch(console.log);
            this.forceUpdate();


  }
 inputChangeHandler = (event) => {
   let formFields = {...this.state.formFields};
   formFields[event.target.name] = event.target.value;
   this.setState({
    formFields
   });


const validateForm = (errors) => {
  let valid = true;
  Object.values(errors).forEach(
    // if we have an error string set valid to false
    (val) => val.length > 0 && (valid = false)
  );
  return valid;
}

const { name, value } = event.target;
     let errors = this.state.errors;
     this.setState({errors, [name]: value}, ()=> {
      console.log(errors)
  })

switch (name) {
       case 'id':
         errors.id =
        value.length < 4
          ? 'Id  must be 4 digits long!'
          : '';
          break;

     }

      if(validateForm(this.state.errors)) {
       console.info('Valid Form')
       this.state.buttonEnabled=false;

  }else{
       console.error('Invalid Form')
       this.state.buttonEnabled=true;
  }

 }
```

```
  render() {
    return(
     <div>
    <div class="panel panel-primary" style={panelStyle}>
      <div class="panel panel-heading">Intern Delete</div>
      <div class="panel panel-body">
        <form onSubmit={this.handleSubmit}>
        <strong>Id:</strong> <br /> <input type="text" name="id"
placeholder="Id" onChange={(e) => this.inputChangeHandler.call(this, e)}
value={this.state.formFields.id} />&nbsp;<span
style={spanStyle}>{this.state.errors.id}</span><br />
 <button class="btn btn-primary" disabled={this.state.buttonEnabled}>Delete
Intern</button>
 </form>
  <p> {this.state.res}</p>

      </div>
    </div>
  </div>

    );
  }
}
export default DeleteInterns
export {DeleteInterns}
```

Test validation,

Unless all inputs are not valid, button will not be enabled.



- View Interns Details
- Register Interns Details
- Update Interns Details
- Update Interns Level
- Delete Interns Details

Intern Registration

Id: 1 — Id must be 4 digits long!

First Name: First Name — First Name cannot be blank

Last Name: Last Name — Last Name cannot be blank

Age: 9 — Age cannot be less than 18

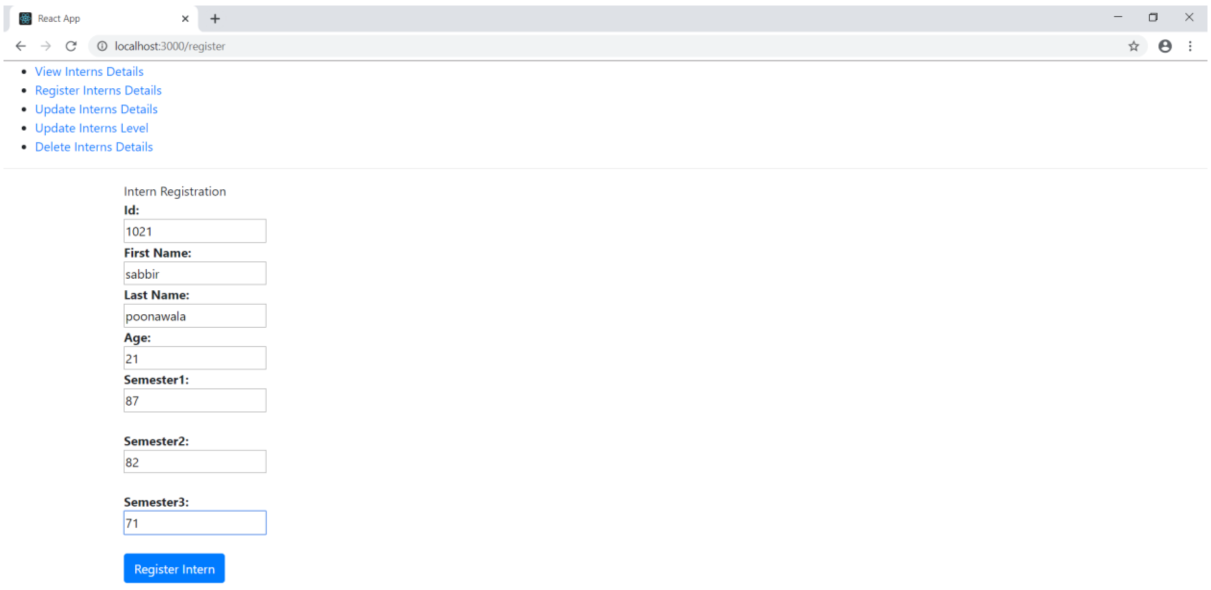
Semester1: -1 — Semester 1 marks cannot be negative

Semester2: -9 — Semester 2 marks cannot be negative

Semester3: -1 — Semester 3 marks cannot be negative

Register Intern

Give valid inputs,



- View Interns Details
- Register Interns Details
- Update Interns Details
- Update Interns Level
- Delete Interns Details

Intern Registration

Id: 1021

First Name: sabbir

Last Name: poonawala

Age: 21

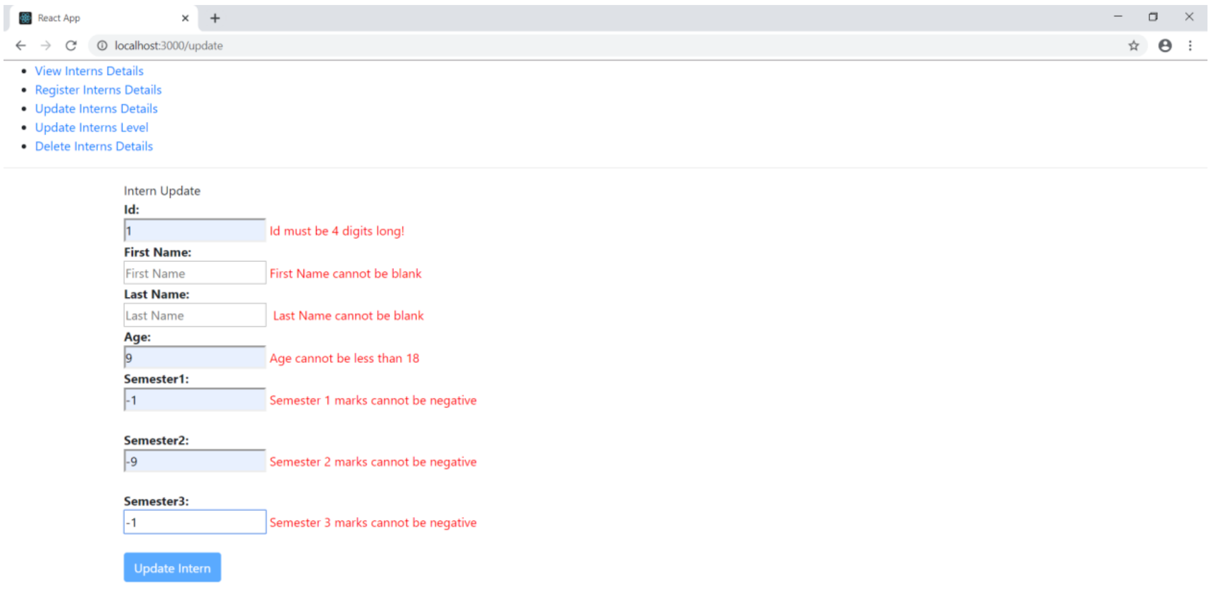
Semester1: 87

Semester2: 82

Semester3: 71

Register Intern

Update Interns,



React App

localhost:3000/update

- View Interns Details
- Register Interns Details
- Update Interns Details
- Update Interns Level
- Delete Interns Details

Intern Update

Id:

1 Id must be 4 digits long!

First Name:

First Name First Name cannot be blank

Last Name:

Last Name Last Name cannot be blank

Age:

9 Age cannot be less than 18

Semester1:

-1 Semester 1 marks cannot be negative

Semester2:

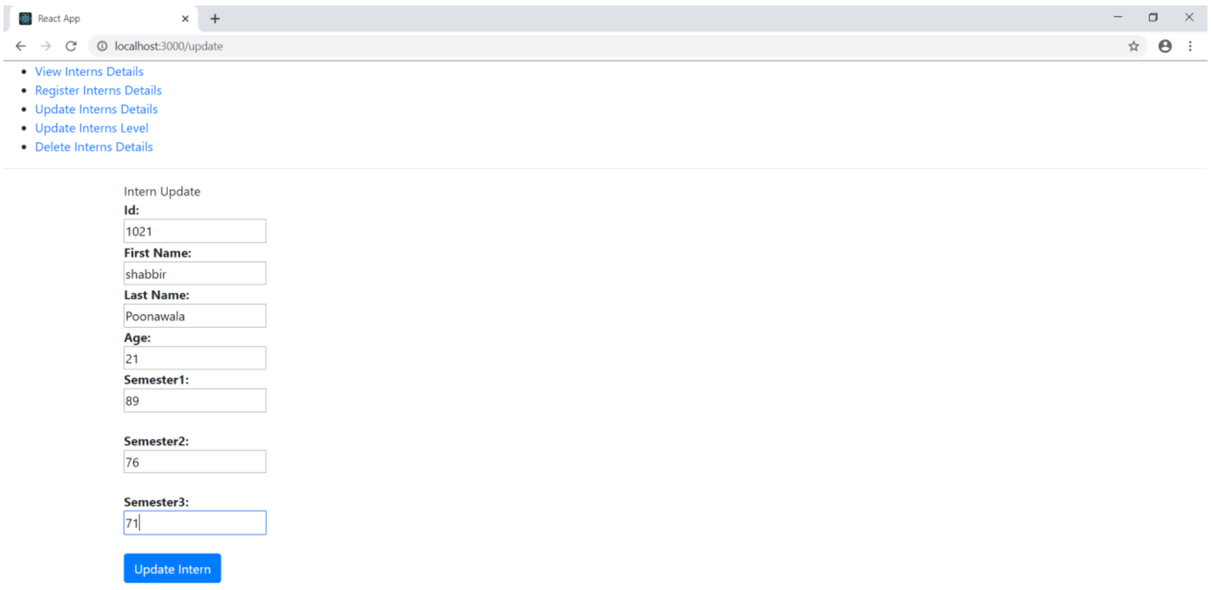
-9 Semester 2 marks cannot be negative

Semester3:

-1 Semester 3 marks cannot be negative

Update Intern

Valid input,



React App

localhost:3000/update

- View Interns Details
- Register Interns Details
- Update Interns Details
- Update Interns Level
- Delete Interns Details

Intern Update

Id:

1021

First Name:

shabbir

Last Name:

Poonawala

Age:

21

Semester1:

89

Semester2:

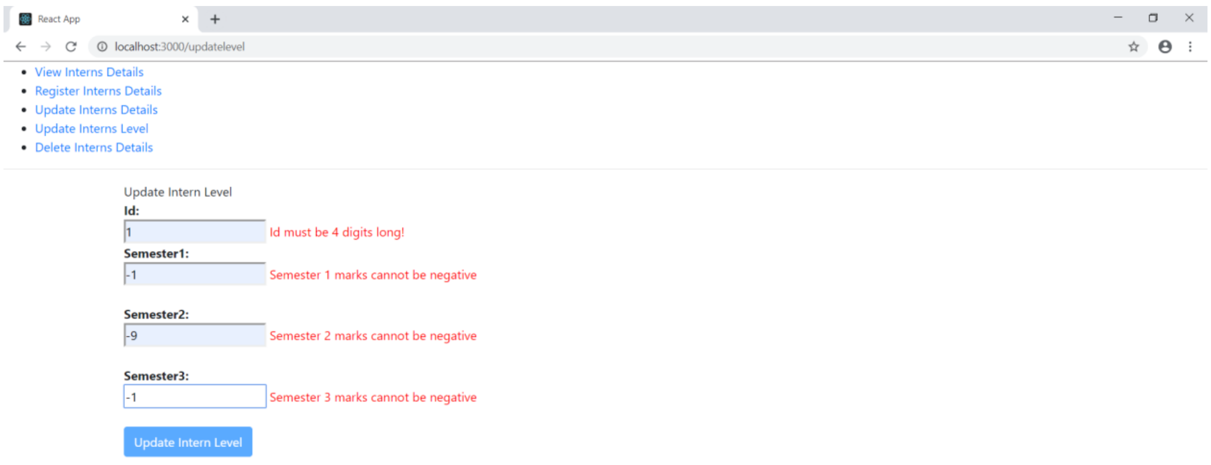
76

Semester3:

71

Update Intern

Update intern level,



React App

localhost:3000/updatelevel

- View Interns Details
- Register Interns Details
- Update Interns Details
- Update Interns Level
- Delete Interns Details

Update Intern Level

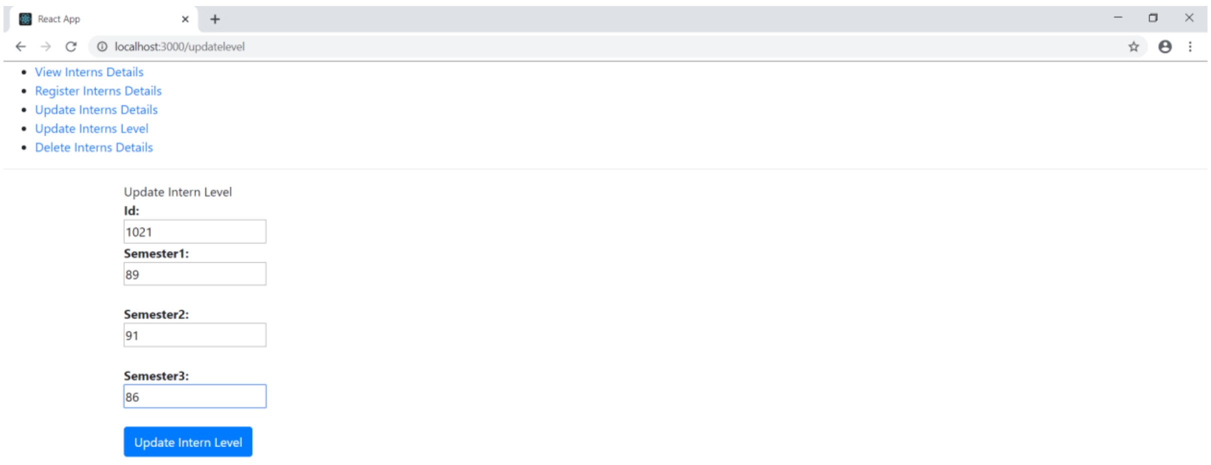
Id: 1 Id must be 4 digits long!

Semester1: -1 Semester 1 marks cannot be negative

Semester2: -9 Semester 2 marks cannot be negative

Semester3: -1 Semester 3 marks cannot be negative

Valid data,



React App

localhost:3000/updatelevel

- View Interns Details
- Register Interns Details
- Update Interns Details
- Update Interns Level
- Delete Interns Details

Update Intern Level

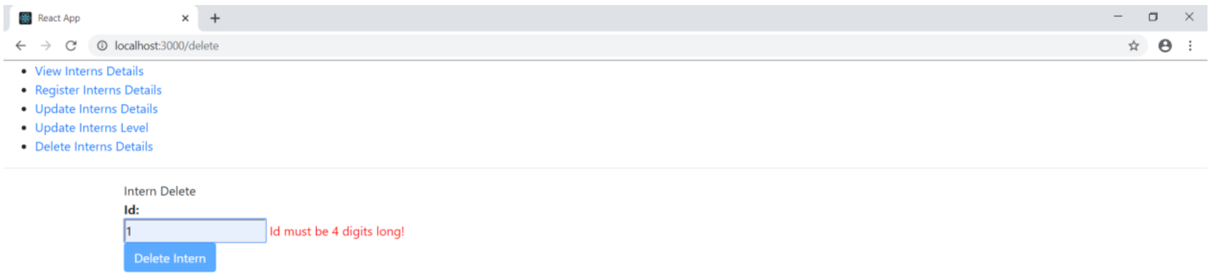
Id: 1021

Semester1: 89

Semester2: 91

Semester3: 86

Delete Intern,


React App

localhost:3000/delete

- View Interns Details
- Register Interns Details
- Update Interns Details
- Update Interns Level
- Delete Interns Details

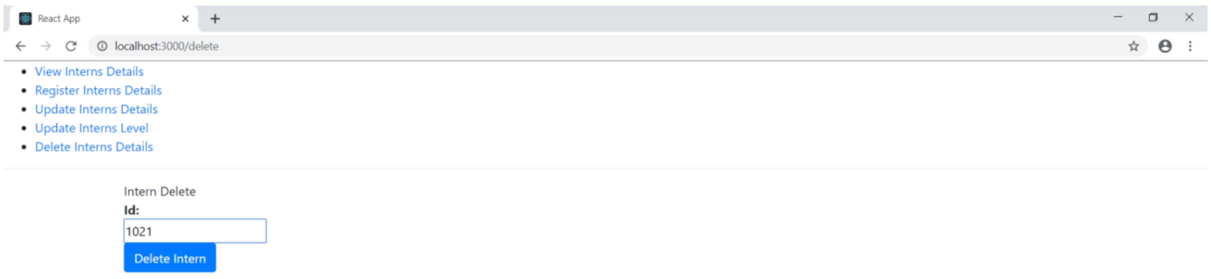
Intern Delete

Id:

1 Id must be 4 digits long!

Delete Intern

Valid intern id,


React App

localhost:3000/delete

- View Interns Details
- Register Interns Details
- Update Interns Details
- Update Interns Level
- Delete Interns Details

Intern Delete

Id:

1021

Delete Intern