**Spring Boot notes**

1. <https://start.spring.io/>
2. Intellij plugin for Spring: Spring Assistant

**Git**

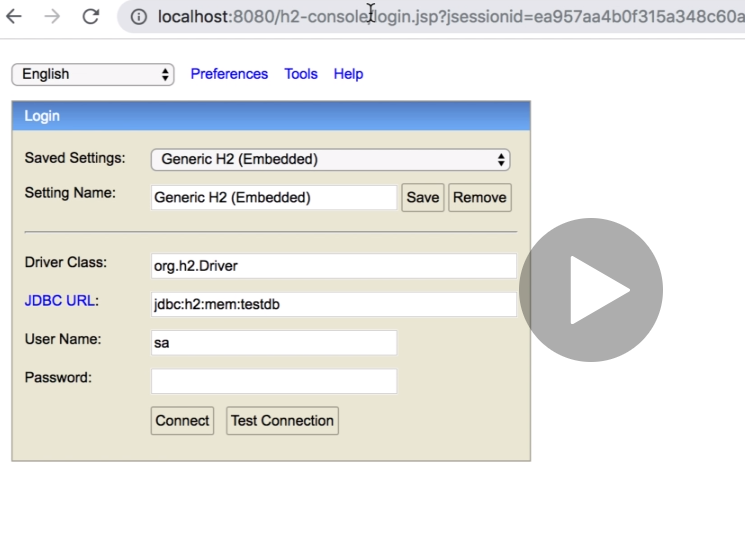
1. Go to github, create project
2. git init
3. git add .
4. git commit –m “first commit”
5. git remote add origin https://github.com/lifukxiang/AgileIntPPMTool.git
6. git push –u origin master
7. git branch <branchname>
8. git checkout <branchname>
9. git merge <branchname>

**IntelJ**

1. Alt + Insert – generate like constructor
2. Ctrl + Alt + S – auto import

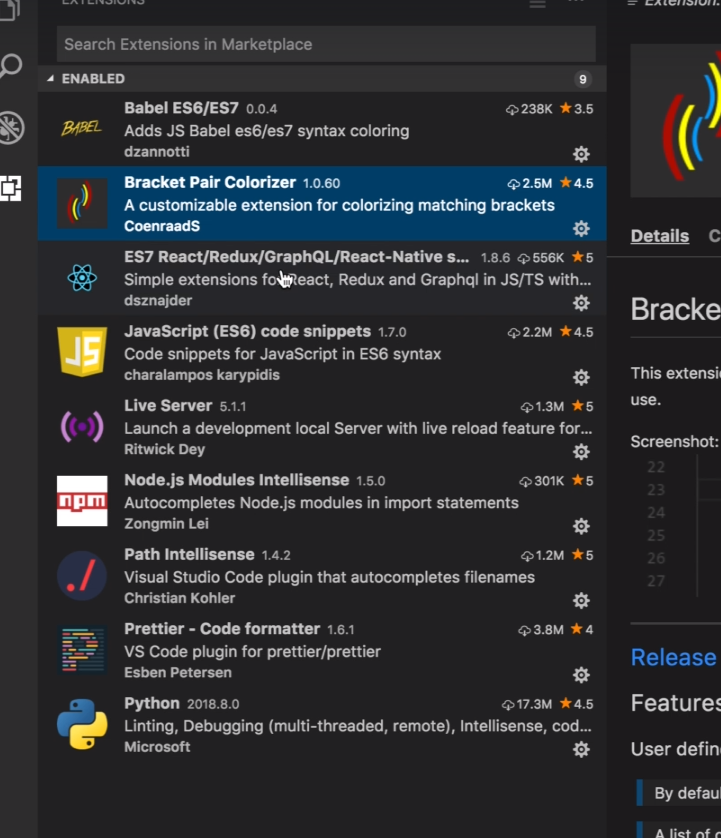
**H2 database**

1. <http://localhost:8080/h2-console>



**VS Code**

1. Extensions



**ReactJS**

1. Install react app using node

npm install –g create-react-app

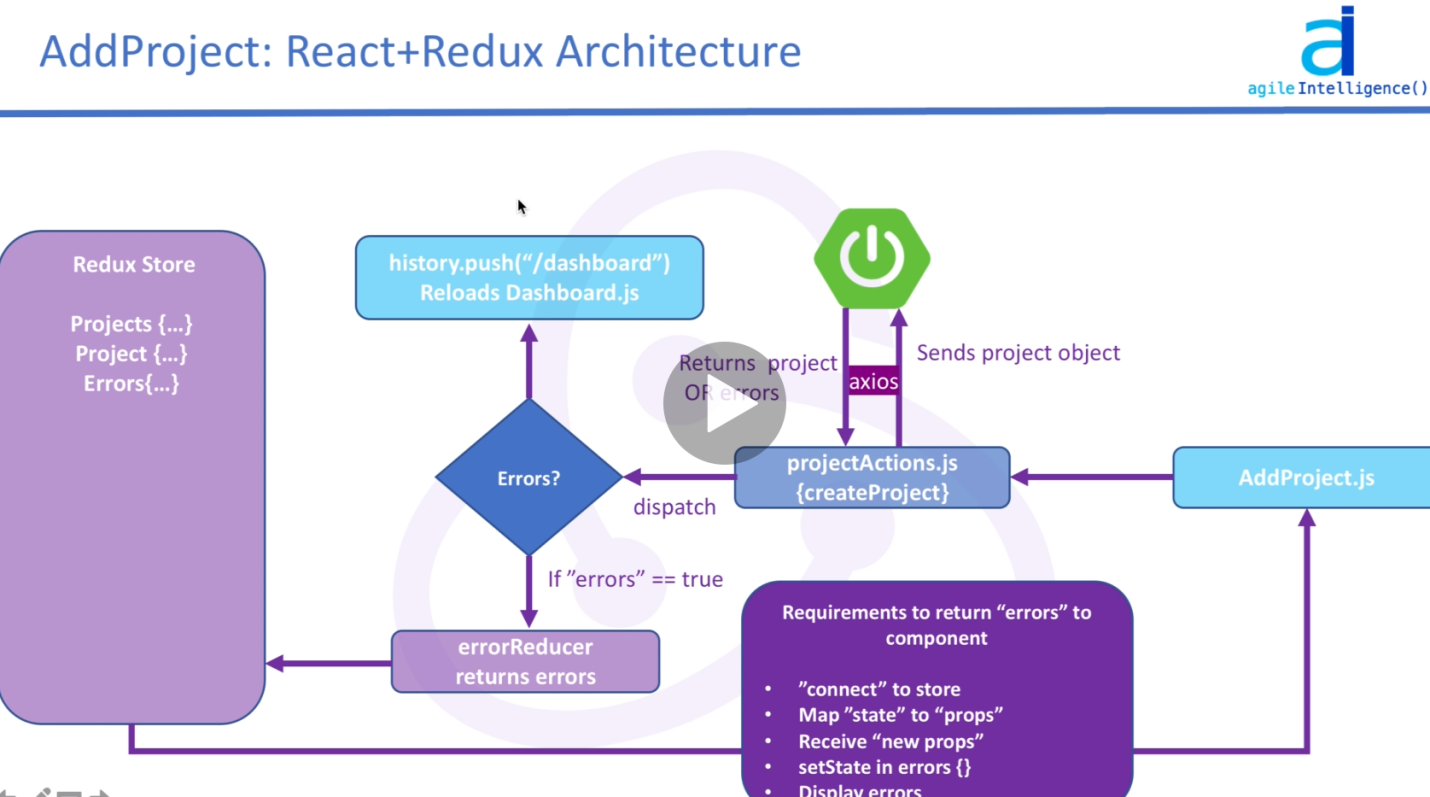
1. Create react app
   1. npx create-react-app my-app or
   2. npm init react-app my-app
2. package.json – scripts, can see what commands to run to start the react application

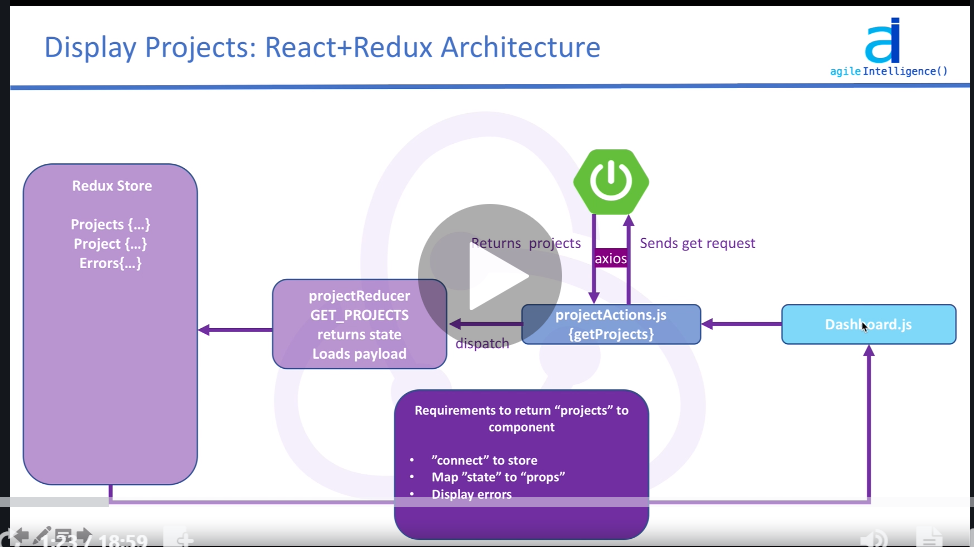
npm start

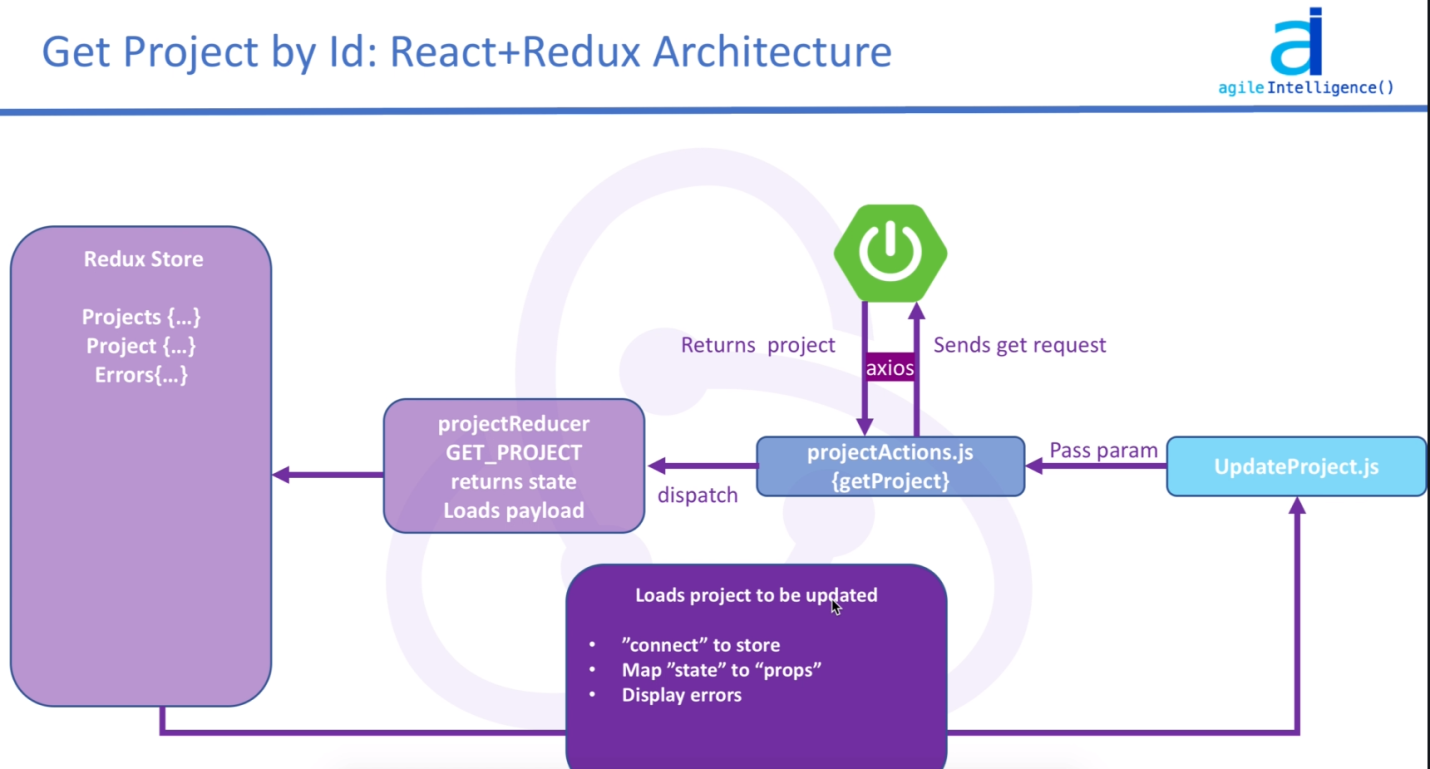
1. Create components
   1. Create a directory: components under src
   2. Create component javascript e.g. dashboard.js
   3. Type rcc in edit
2. App.js is main js
   1. With plugin installed earlier, we can easily render the component by doing e.g. <Dashboard />
3. Install bootstrap
   1. npm install bootstrap
   2. to use bootstrap, import it in App.js

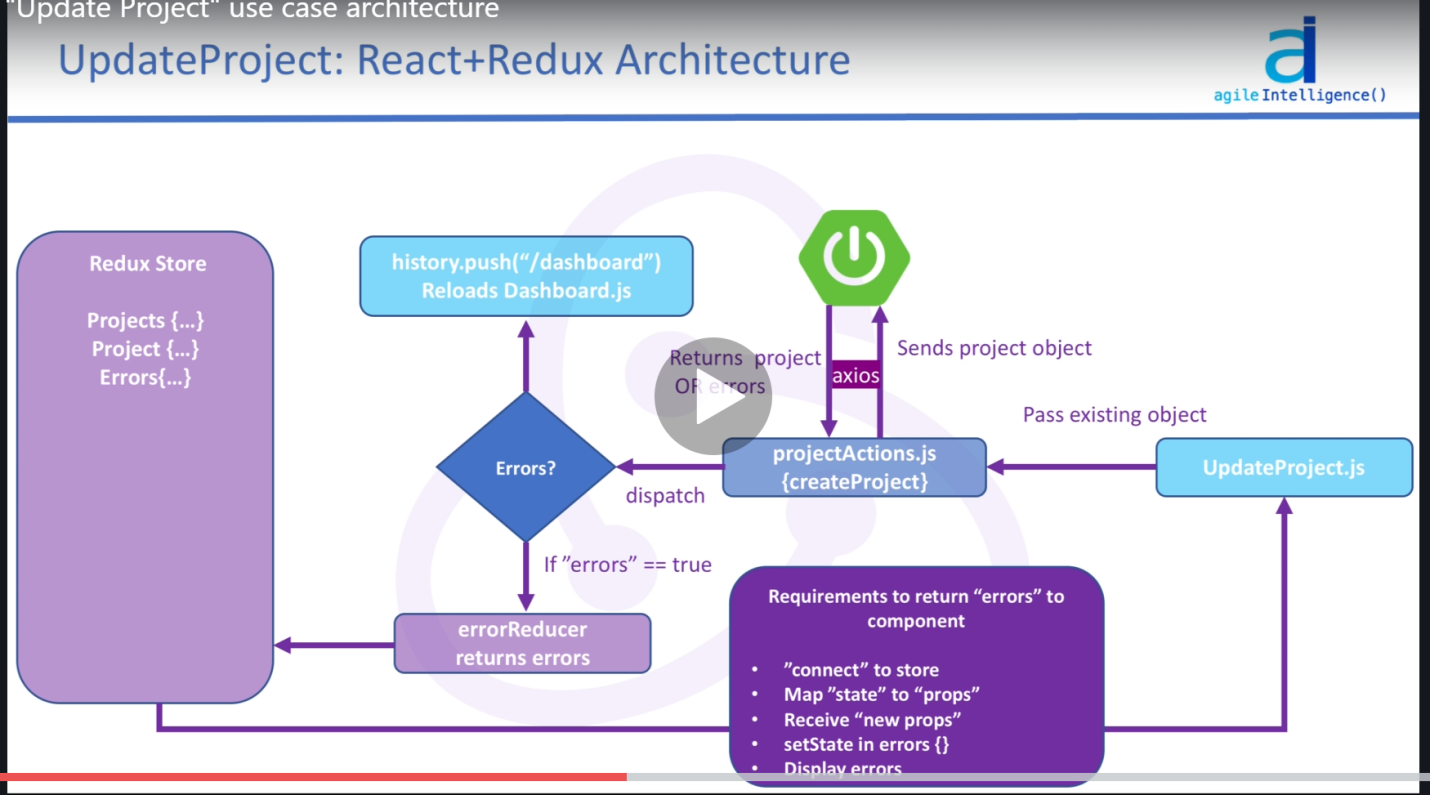
import "bootstrap/dist/css/bootstrap.min.css";

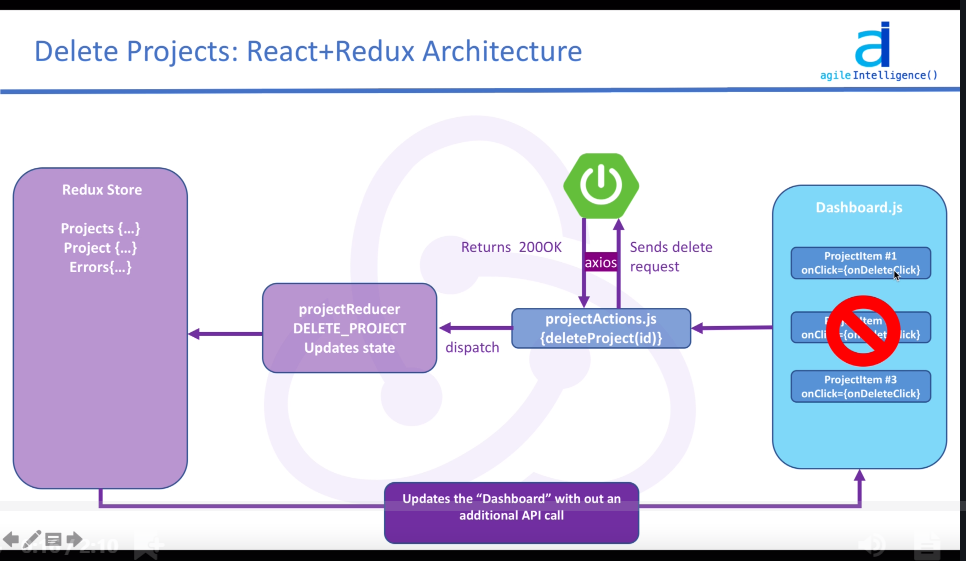
1. Tips
   1. in editor, control D to select all element like class, and type to change to className
   2. sample functional component e.g. CreateProjectButton.js
   3. a href, replaced with {Link} component, from react-router-dom
   4. React.Fragment to replace div
   5. React-redux architecture











1. Include font awesome, go to index.html and include below link

<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.2.0/css/all.css" integrity="sha384-hWVjflwFxL6sNzntih27bfxkr27PmbbK/iSvJ+a4+0owXq79v+lsFkW54bOGbiDQ"

crossorigin="anonymous">

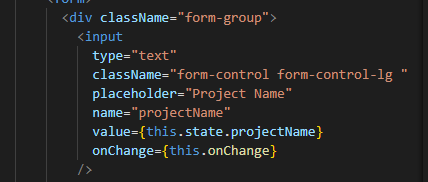
1. React routing
   1. To install react router dom, npm install react-router-dom
   2. Go to App.js, import

import { BrowserRouter as Router, Route } from "react-router-dom";

* 1. Different routing to different component, define in App.js e.g.

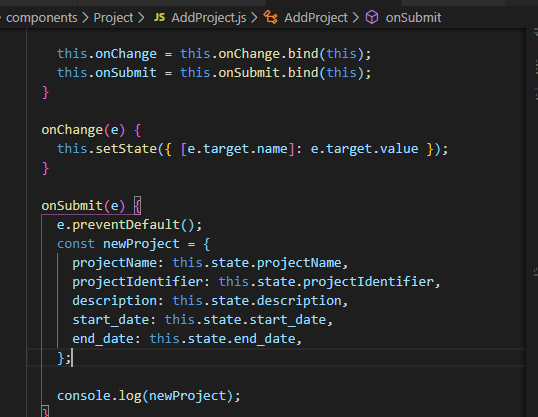
1. <Router>
2. <div className="App">
3. <Header />
4. <Route exact path="/dashboard" component={Dashboard} />
5. <Route exact path="/addProject" component={AddProject} />
6. </div>
7. </Router>
8. Functional Component
   1. Refer to AddProject.js
   2. Design the html from scratch
   3. Copy to render method
   4. Tips share
9. //check name attribute input fields
10. //create constructor
11. //set state
12. //set value on input fields
13. //create onChange function
14. //set onChange on each input field
15. //bind on constructor
16. //check state change in the react extension
17. Name shall match api entity name
18. Copy from postman for those field
19. Binding input to the state on change

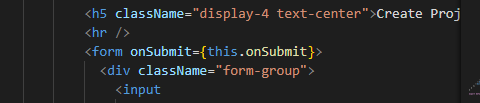




1. Binding on submit for form

e.preventSubmit(); is to preserve form input, instead of refreshing the form.





**Redux**

1. Install redux libraries: npm i redux react-redux redux-thunk
2. Install axios for get, post: npm i axios
3. Setting up and creating store

|  |
| --- |
| 1. Create store.js on root of the folder   import { createStore, applyMiddleware, compose } from "redux";  import thunk from "redux-thunk";  import rootReducer from "./reducers";  const initialState = {};  const middleware = [thunk];  let store;  if (window.navigator.userAgent.includes("Chrome")) {    store = createStore(      rootReducer,      initialState,      compose(        applyMiddleware(...middleware),        window.\_\_REDUX\_DEVTOOLS\_EXTENSION\_\_ &&          window.\_\_REDUX\_DEVTOOLS\_EXTENSION\_\_()      )    );  } else {    store = createStore(      rootReducer,      initialState,      compose(applyMiddleware(...middleware))    );  }  export default store;   1. Create reducers folder & index.js inside the folder   import { combineReducers } from "redux";  export default combineReducers({});   1. In App.js, import Provider and store   import { Provider } from "react-redux";  import store from "./store";  function App() {    return (      <Provider store={store}>        <Router>          <div className="App">            <Header />            <Route exact path="/dashboard" component={Dashboard} />            <Route exact path="/addProject" component={AddProject} />          </div>        </Router>      </Provider>    );  } |

**React Action, Reducer**

|  |
| --- |
| 1. Define types.js under action folder   export const GET\_ERRORS = "GET\_ERRORS";   1. Update errorReducer   import { GET\_ERRORS } from "../actions/types";  const initialState = {};  export default function (state = initialState, action) {    switch (action.type) {      case GET\_ERRORS:        return action.payload;      default:        return state;    }  }   1. Update projectActions.js   import axios from "axios";  import { GET\_ERRORS } from "./types";  const API\_HOST = "http://localhost:8080";  export const createProject = (project, history) => async (dispatch) => {    try {      const res = await axios.post(API\_HOST + "/api/project", project);      history.pushState("/dashboard");    } catch (err) {      dispatch({        type: GET\_ERRORS,        payload: err.response.data,      });    }  };   1. AddProject.js – connect to redux, and invoke action   import PropTypes from "prop-types";  import { connect } from "react-redux";  import { createProject } from "../../actions/projectActions";    onSubmit(e) {      e.preventDefault();      const newProject = {        projectName: this.state.projectName,        projectIdentifier: this.state.projectIdentifier,        description: this.state.description,        start\_date: this.state.start\_date,        end\_date: this.state.end\_date,      };      console.log(newProject);      this.props.createProject(newProject, this.props.history);    }  AddProject.propTypes = {    createProject: PropTypes.func.isRequired,  };  export default connect(null, { createProject })(AddProject); |

**Error Reducer & validation error**

|  |
| --- |
| 1. **Define errors**     constructor() {      super();      this.state = {        projectName: "",        projectIdentifier: "",        description: "",        start\_date: "",        end\_date: "",        errors: {},      };      this.onChange = this.onChange.bind(this);      this.onSubmit = this.onSubmit.bind(this);    }   1. **Define propTypes for error**   AddProject.propTypes = {    createProject: PropTypes.func.isRequired,    errors: PropTypes.object.isRequired,  };   1. **Map state to props**   const mapStateToProps = (state) => ({    errors: state.errors,  });   1. **Connect to redux**   export default connect(mapStateToProps, { createProject })(AddProject);   1. **Render errors**     render() {      const { errors } = this.state;      return (        <div>          {            //check name attribute input fields            //create constructor            //set state            //set value on input fields            //create onChange function            //set onChange on each input field            //bind on constructor            //check state change in the react extension          }          <div className="register">            <div className="container">              <div className="row">                <div className="col-md-8 m-auto">                  <h5 className="display-4 text-center">Create Project form</h5>                  <hr />                  <form onSubmit={this.onSubmit}>                    <div className="form-group">                      <input                        type="text"                        className="form-control form-control-lg "                        placeholder="Project Name"                        name="projectName"                        value={this.state.projectName}                        onChange={this.onChange}                      />                      <p>{errors.projectName}</p>   1. **Beautify**   **npm I classname**  import classnames from "classnames";    <input                        type="text"  **className={classnames("form-control form-control-lg", {**  **"is-invalid": errors.projectName,**  **})}**                        placeholder="Project Name"                        name="projectName"                        value={this.state.projectName}                        onChange={this.onChange}                      />                      {errors.projectName && (                        <div className="invalid-feedback">                          {errors.projectName}                        </div>                      )} |

**Spring Security, JWT**

1. Include maven dependency

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-security</artifactId>  
</dependency>  
<dependency>  
 <groupId>com.google.code.gson</groupId>  
 <artifactId>gson</artifactId>  
 <version>2.8.5</version>  
</dependency>  
<dependency>  
 <groupId>io.jsonwebtoken</groupId>  
 <artifactId>jjwt</artifactId>  
 <version>0.9.0</version>  
</dependency>

1. Important links
   1. <https://spring.io/guides/topicals/spring-security-architecture/>
   2. <https://spring.io/guides/gs/securing-web/>
   3. <https://auth0.com/blog/implementing-jwt-authentication-on-spring-boot/>

React Security

1. npm install jwt-decode – to decode jwt token
2. rfc – another shortcut to create function