Software Design and Test Specification

SiderProject

(Project Management and Employee Scheduling application)

by Steven Curran

Student ID: 20235873

NUIG Supervisor: Adrian Clear

Sidero Supervisor: Sourav Bhattacharya

Github link:

https://github.com/StevenCurranBlacklion/SiderProject

Table of Contents

1. Introduction

1.1 Document Outline

The main aim of this document is to introduce you, the reader, to the project. It will explain how the application was developed, how it works, what tools were used and whether it did or didn’t fulfil the main requirements through various tests. This report will also document the students learning outcomes from doing the project.

1.2 Document Description

1.2.1 Introduction

For this project, the student has created a full stack application called “SiderProject”.

What is SiderProject?

SiderProject is a project management and employee scheduling application that has the ability to create projects, set tasks for the project and assign employees with certain skills to the project.

The inspiration to do a full stack application came from the advisor of the students internship, Sidero. The student was also encouraged to use similar technologies as the company so through practice of them, they are all already training themselves before the job even begins. These technologies will be elaborated on further down in the Core Technologies section.

1.2.2 System Overview

As stated above, SiderProject is a full stack application which makes use of a Java Spring backend, ReactJS frontend and H2 database. The system is designed in mind for those working in companies to keep track of their various projects. The project portion alone can also be used by freelance developers to keep track of their own projects for their clients. The system is secure requiring a login and has differences between an admin and user login. It is easy to use and very user friendly. SiderProject has the ability to create unique projects and employees with specific tasks and skills on the click of a button. The actual architecture of the system will be elaborated on further below.

1.2.3 Main Requirements

The following are the main requirements of the app:

- The ability to add, edit and delete projects.
- The ability to add, edit and delete employees.
- The ability to assign employees to a specific project.
- The ability to add, edit and delete tasks to a specific project.
- The ability to add, edit and delete skills to a specific employee.
- The ability to have multiple logins who can access the same data.
- The ability to have some accounts be admin who can add, edit, delete while non-admins can only view.

2. Design Considerations

2.1 General Constraints

A number of constraints were adhered to during the development either by the students personal choice or by recommendation of either their internship advisor or university advisor.

The most obvious constraint is the restriction of technologies used to those that the students internship Sidero share. So the student was heavily encouraged to use Java Spring with Maven as Sidero are a Java heavy organization. They were also encouraged to use ReactJS as the framework as this is the framework primarily used by Sidero. Coding standards and best practices were adhered to as best as possible as well.

The student also tried to make the app as user friendly as possible so in other words, avoid a design and user interface which could be considered complicated.

2.2 Goals

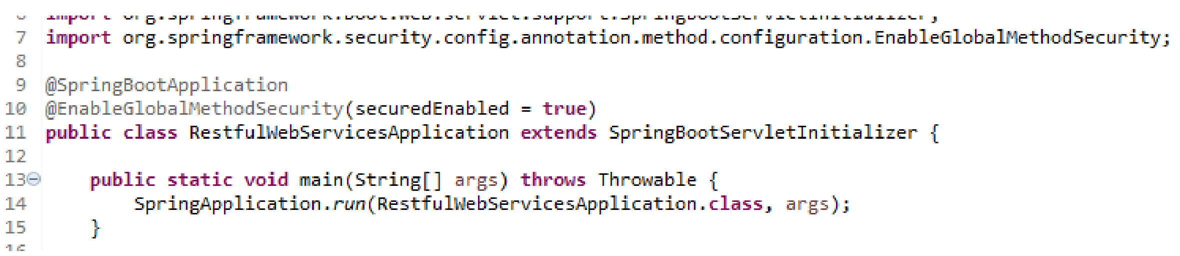
The following were the goals the student had for the project:

- To provide a user friendly app which is easy to use.
- To provide an app which stands out from competitors.
- To develop an app with the 4 basic operations of create, read, update and delete.
- To develop an app which fulfils the bare minimum requirements.
- To develop an app which includes all desirable (but not essential) features.

3. Detailed System Design

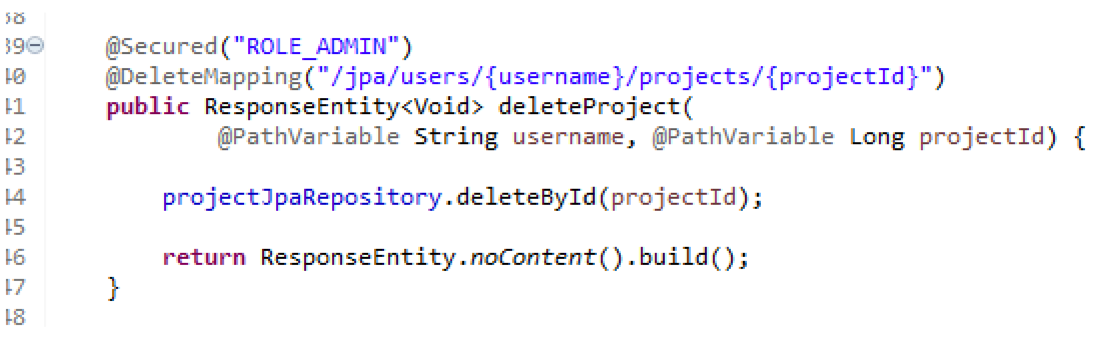
3.1 Use Case Documentation

As can be seen in the two Use Case Diagrams below, there are two different types of actors which can access the system. The Admin actor has access to everything in particular the add, update and delete functions. The User actor can view everything but cannot add, update or delete any projects. By adding @EnableGlobalSecurity to the RestfulWebServicesApplication in the Java backend and placing @Secured("ROLE_ADMIN") above the mappings the student wants restricted from non-admins, this can be achieved.



```java
import org.springframework.security.config.annotation.method.configuration.EnableGlobalMethodSecurity;

@SpringBootApplication
@EnableGlobalMethodSecurity(securedEnabled = true)
public class RestfulWebServicesApplication extends SpringBootServletInitializer {

    public static void main(String[] args) throws Throwable {
        SpringApplication.run(RestfulWebServicesApplication.class, args);
    }
```

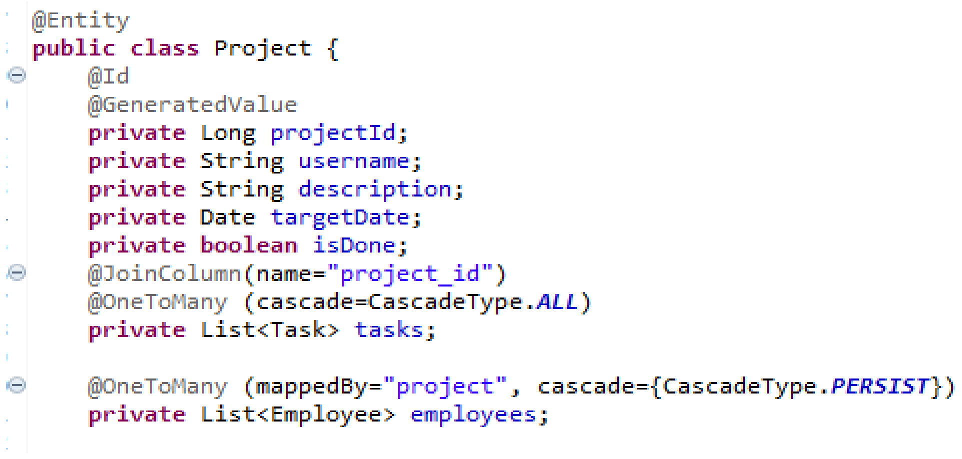
Figure 1: @EnableGlobalSecurity



```java
    @Secured("ROLE_ADMIN")
    @DeleteMapping("/jpa/users/{username}/projects/{projectId}")
    public ResponseEntity<Void> deleteProject(
            @PathVariable String username, @PathVariable Long projectId) {

        projectJpaRepository.deleteById(projectId);

        return ResponseEntity.noContent().build();
    }
```

Figure 2: @Secured("ROLE_ADMIN")

Using the @OneToMany and @JoinColumns annotations, the student can connect Projects, Employees, Tasks and Skills to each other in the H2 database.



```java
@Entity
public class Project {
    @Id
    @GeneratedValue
    private Long projectId;
    private String username;
    private String description;
    private Date targetDate;
    private boolean isDone;
    @JoinColumn(name="project_id")
    @OneToMany (cascade=CascadeType.ALL)
    private List<Task> tasks;

    @OneToMany (mappedBy="project", cascade={CascadeType.PERSIST})
    private List<Employee> employees;
```

Figure 3: Connect Tables

The Projects section is accessed from the home page. Tasks are then accessed from the Projects section by clicking the View Tasks button on a specific project. As seen in the Use Case Diagram there is both a View Employees and Assigned Employees Use Case. View Employees (where you view all employees) are accessed from the home page while Assigned Employees (employees assigned to specific project) are accessed from the projects section by clicking the View Employees button. Skills are accessed from either Employees section by clicking the View Skills button on a specific employee. In terms of which table has which foreign key:

- Project ID is a foreign key of the Employee and Tasks tables.
- Employee ID is a foreign key of the Skills table.
- Task ID is also a foreign key of the Employees table.

3.2 Use Case Diagrams

<u>Admin</u>

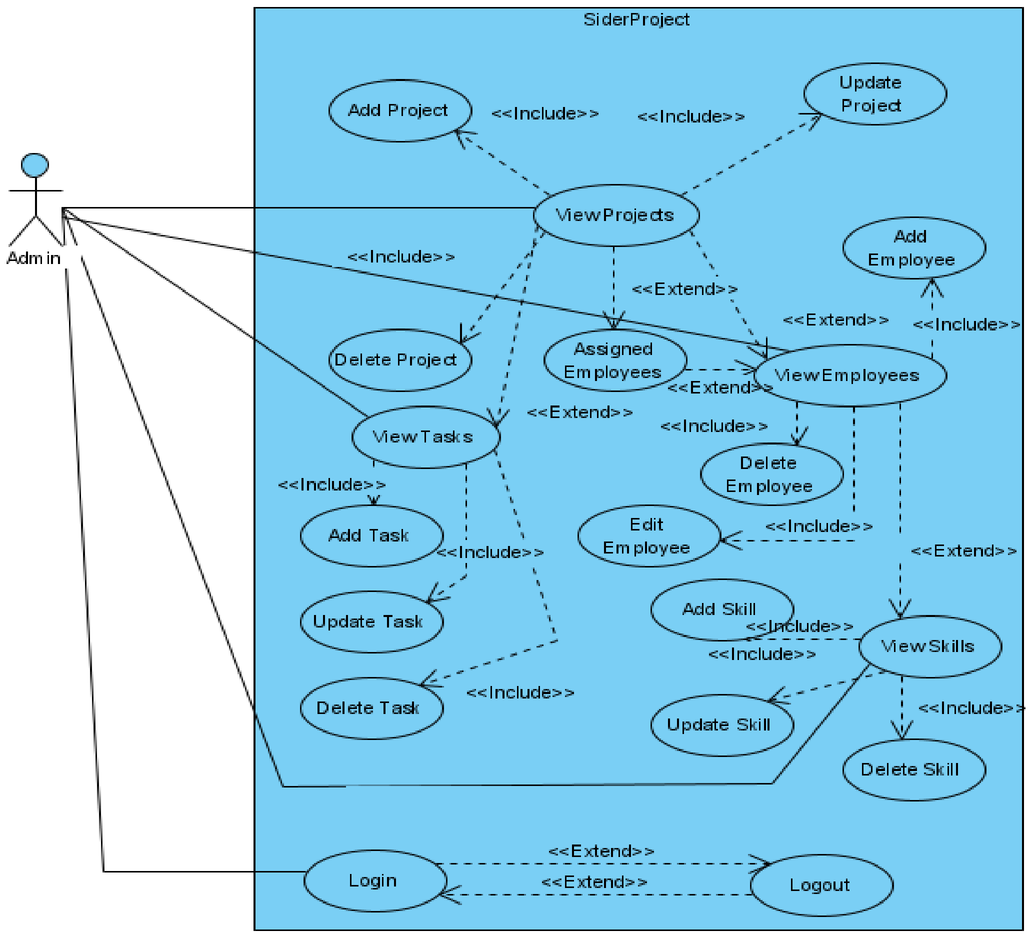


Figure 4: Admin Use Case Diagram

User

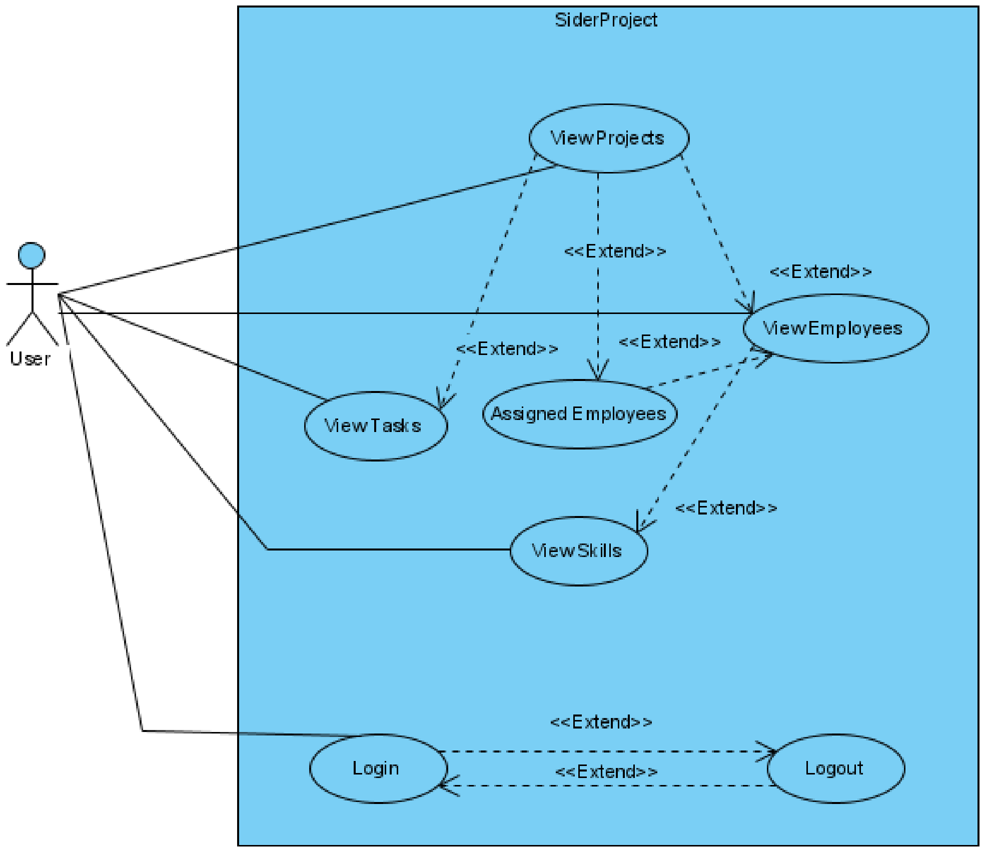


Figure 5: User Use Case Diagram

I have created Use Case Descriptions for all the use cases shown above. Below I will discuss the View, Add, Update and Delete Project Use Case Descriptions which act as counterparts to the equivalents for Employees, Tasks and Skills. Login and Logout also speak for themselves in the Use Case Diagrams above. The remaining Use Case Descriptions can be seen in a pdf link in Appendix 3.

3.3 Use Case Descriptions

The view Projects Use Case below and by extension the View Employees, Skills, Tasks and Assigned Employees all operate in a similar manner where using a simple @GetMapping annotation, we can display data from a H2 database table in the database. The Assigned Employees, View Tasks and View Skills get method will only display the items with the relevant foreign key (Project ID or Employee ID).

Use case name:	View Projects
Description	The actor wants to view the projects section.
Participating actors	Admin, User
Entry conditions	The actor needs to be logged in and at the home page.
Flow of events	1. Click on the projects tab to see current projects. 2. Choose CRUD option. See steps 1 of Add Project, Update Project or Delete Project (Admin only).
Alternative flow of events	1. After step 1, actor has the option to view Employees or Tasks in order to see Employees / Tasks assigned to project. Continues from step 1 of Assigned Employees or View Tasks.
Exceptional flow of events	1. In step 2, if actor is not admin, option will fail.
Relationships	Includes: • Add Project • Update Project • Delete Project Extended by: • Assigned Employees • View Tasks

The Add Project Use Case below and its Employee, Task and Skill counterparts use the @PostMapping annotation to create the relevant object to add to the H2 database table for display.

Use case name:	Add Project
Description	The actor wants to add a project to the projects list.
Participating actors	Admin
Entry conditions	The admin needs to be logged in and have selected the projects tab.
Flow of events	1. Click on the Add Project button 2. Enter valid details. 3. Click save
Alternative flow of events	N/A
Exceptional flow of events	1. In step 1, if actor is not admin, error will pop up. 2. In step 3, if details entered were not valid, admin is prompted to enter again.
Relationships	Extends: • View Projects

The Update Project Use Case below and its Employee, Task and Skill counterparts use the @PutMapping annotation in order to update the selected object. This can include the change of foreign keys to assign it to a different displayed list or the setting of foreign keys to null such as Employees which will display it in the main Employees list but not be assigned to any project yet.

Use case name:	Update Project
Description	The admin wants to update a current project.
Participating actors	Admin
Entry conditions	The admin needs to be logged in and have selected the projects tab.
Flow of events	1. Click on the Update Project button 2. Change details. 3. Click save
Alternative flow of events	N/A
Exceptional flow of events	1. In step 1, if actor is not admin, error will pop up. 2. In step 3, if details entered were not valid, admin is prompted to enter again.
Relationships	Extends: • View Projects

The Delete Project Use Case below and its Employee, Task and Skill counterparts use the @DeleteMapping annotation in order to delete the selected object. When deleting a Project specifically, this will take all assigned Tasks to it with it but will only change assigned employees foreign key (Project ID) to null. This is because Employees has its own separate list displaying all from the table and they are not completely tied to Projects like Tasks are. Employees and Skills operate in the same manner as Projects and Tasks.

Use case name:	Delete Project
Description	The admin wants to delete a current project.
Participating actors	Admin
Entry conditions	The admin needs to be logged in and have selected the projects tab.
Flow of events	1. Click on the Delete Project button
Alternative flow of events	N/A
Exceptional flow of events	1. In step 1, if actor is not admin, error will pop up.
Relationships	Extends: • View Projects

4. Core Technologies

4.1 Java Spring / Spring Boot

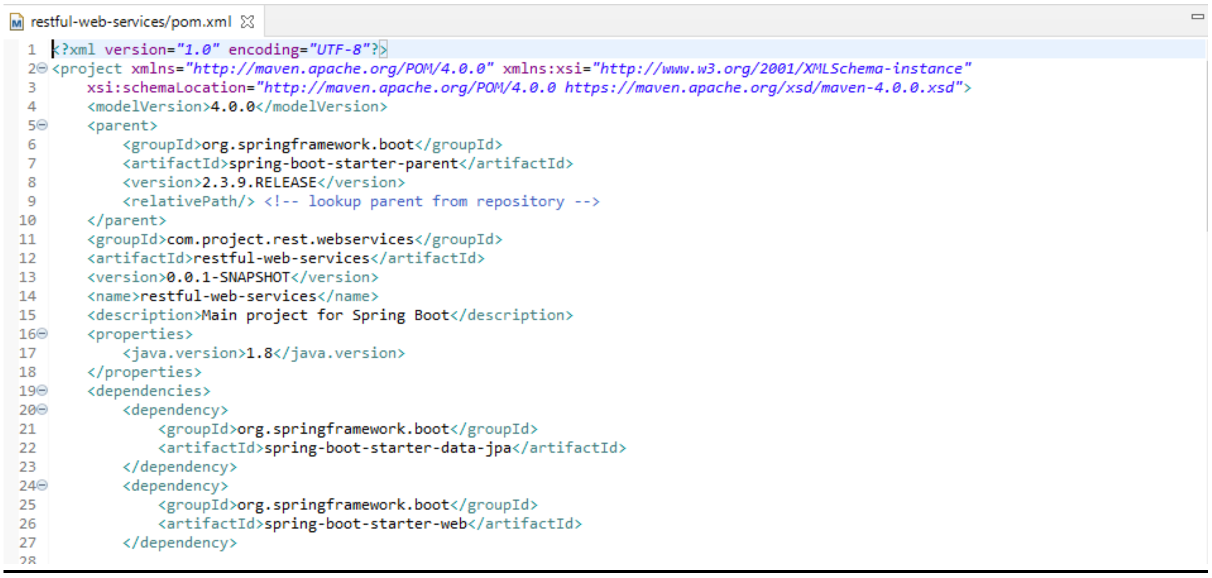
Java Spring was the technology which encompassed the backend of the project. Spring is an application framework for the Java platform. The student had downloaded a Maven project from Spring Initializr and worked from there. The Maven Project was then imported into the Eclipse IDE which was the IDE used throughout the development of the backend of the project.

What is Maven?

Maven is a build automation tool. In other words, Maven is what was used to build and manage the Java Spring portion (backend) of the project.

4.1.1 Dependencies

Java Spring has various "dependencies". Dependencies can be added either at inception when generating the Maven project from the Spring Initializr or at any point in the pom.xml file as seen below.



restful-web-services/pom.xml

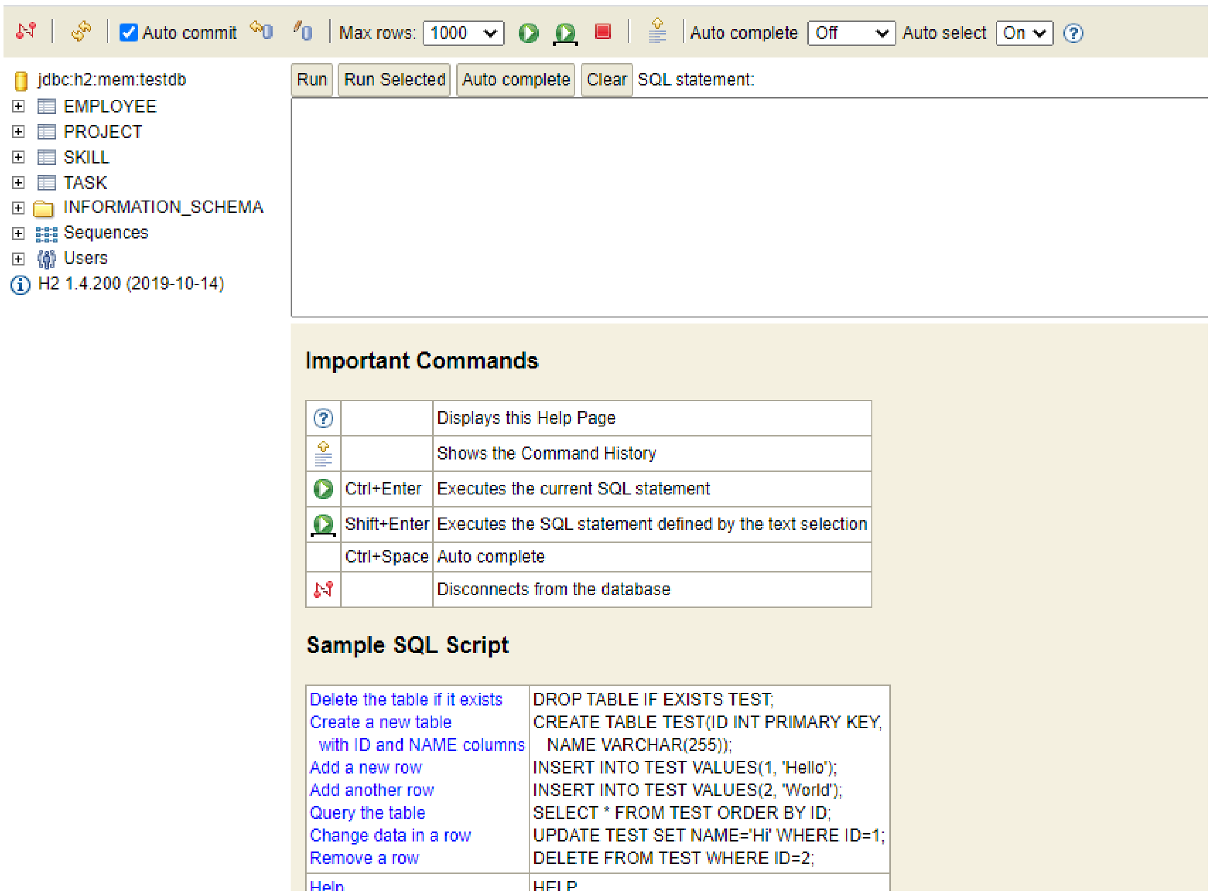
```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.3.9.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.project.rest.webservices</groupId>
    <artifactId>restful-web-services</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>restful-web-services</name>
    <description>Main project for Spring Boot</description>
    <properties>
        <java.version>1.8</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
```

Figure 6: pom.xml

Dependencies that the student added were Spring Web, Spring Security, JPA, Devtools and H2. H2 specifically was the inbuilt database used for the project.

4.1.2 H2 Database

As stated above, the database chosen for the app was a H2 database which is a dependency of Java Spring. Although the student had considered using a MySQL database from PHPMyAdmin which is what he learned in his databases module, the student went with the H2 database for simple convenience since the database management system is written in java and automatically connected to the backend. Below is the appearance of the H2 database in the application.



Important Commands

?		Displays this Help Page
		Shows the Command History
	Ctrl+Enter	Executes the current SQL statement
	Shift+Enter	Executes the SQL statement defined by the text selection
	Ctrl+Space	Auto complete
		Disconnects from the database

Sample SQL Script

Delete the table if it exists	DROP TABLE IF EXISTS TEST;
Create a new table with ID and NAME columns	CREATE TABLE TEST(ID INT PRIMARY KEY, NAME VARCHAR(255));
Add a new row	INSERT INTO TEST VALUES(1, 'Hello');
Add another row	INSERT INTO TEST VALUES(2, 'World');
Query the table	SELECT * FROM TEST ORDER BY ID;
Change data in a row	UPDATE TEST SET NAME='Hi' WHERE ID=1;
Remove a row	DELETE FROM TEST WHERE ID=2;
Help	HELP

Figure 7: H2 Database

4.1.3 JPA and Hibernate

JPA was the API used in the application making it one of the most important dependencies of Java Spring. JPA is what allows the classes to map to the tables in the H2 database. The type of classes which map to the tables are called "JPA entities". These will be elaborated on further below in System

Architecture. One of JPA’s many implementations is Hibernate. Hibernate is an object to relation mapping framework. This was used throughout development of the backend to determine relations between the various classes. As stated further above in the detailed system design, this included annotations such as @OneToMany and @JoinColumns which would make one class object a foreign key of another class.

4.2 ReactJS

ReactJS was the framework used to create the User Interface of the application. The IDE used throughout was Visual Studio Code. While the Java Spring backend and H2 database represented all of the inner workings of the application, the ReactJS UI allowed these functionalities to display and be used by an actor. Compared to Java in the backend, the student had no prior experience with ReactJS going into the project. Only standard javascript so a lot of this was a new learning experience for the student. How exactly the ReactJS frontend interacted with the backend will be explained below in System Architecture.

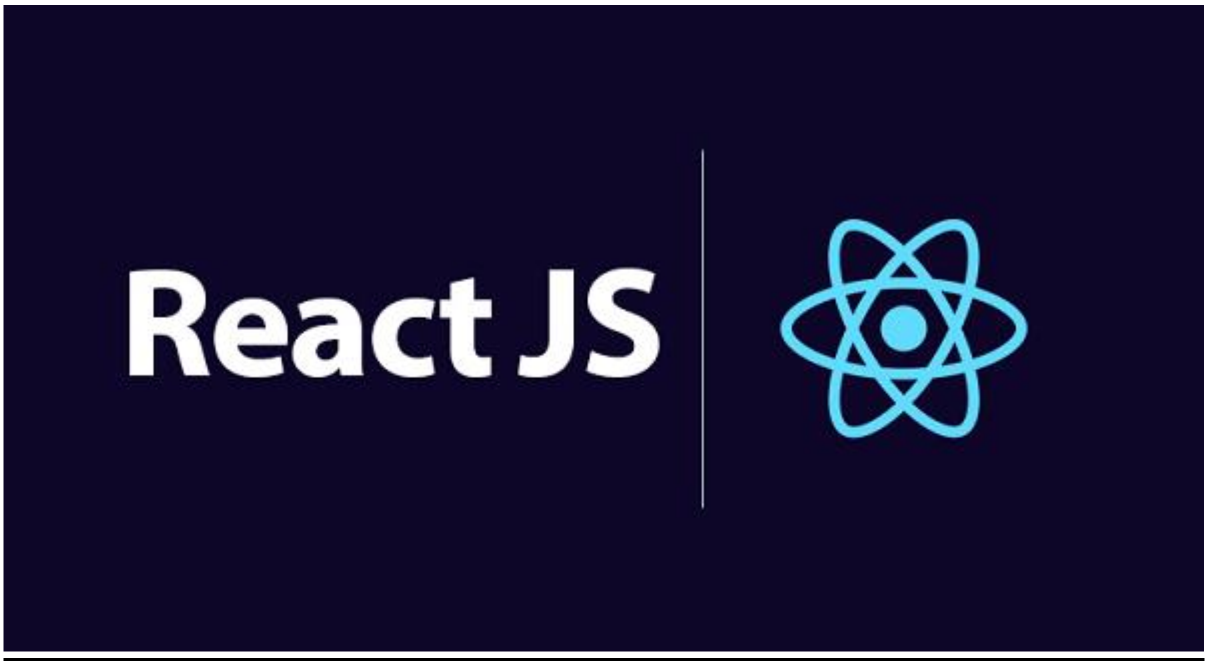
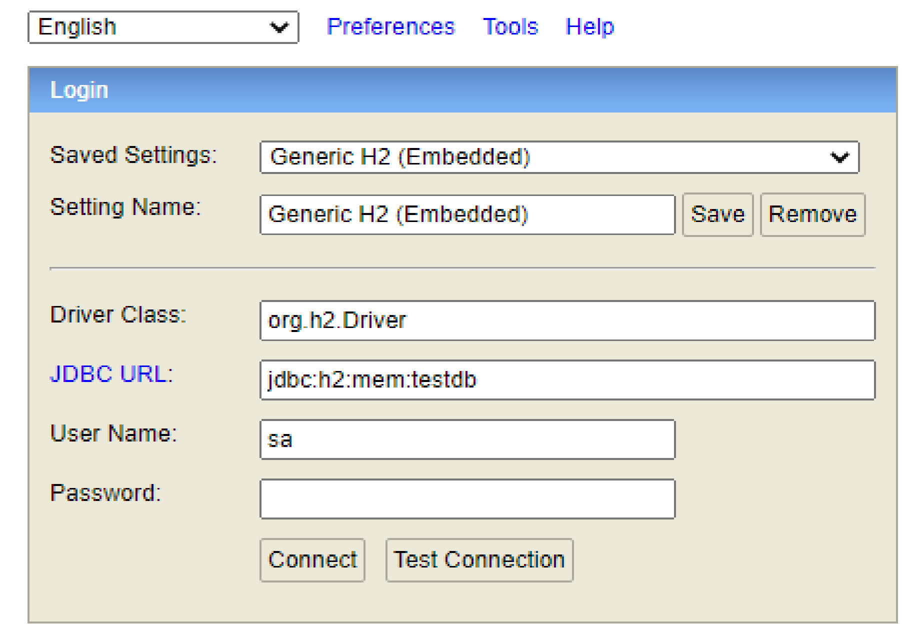


Figure 8: ReactJS logo

5. System Architecture

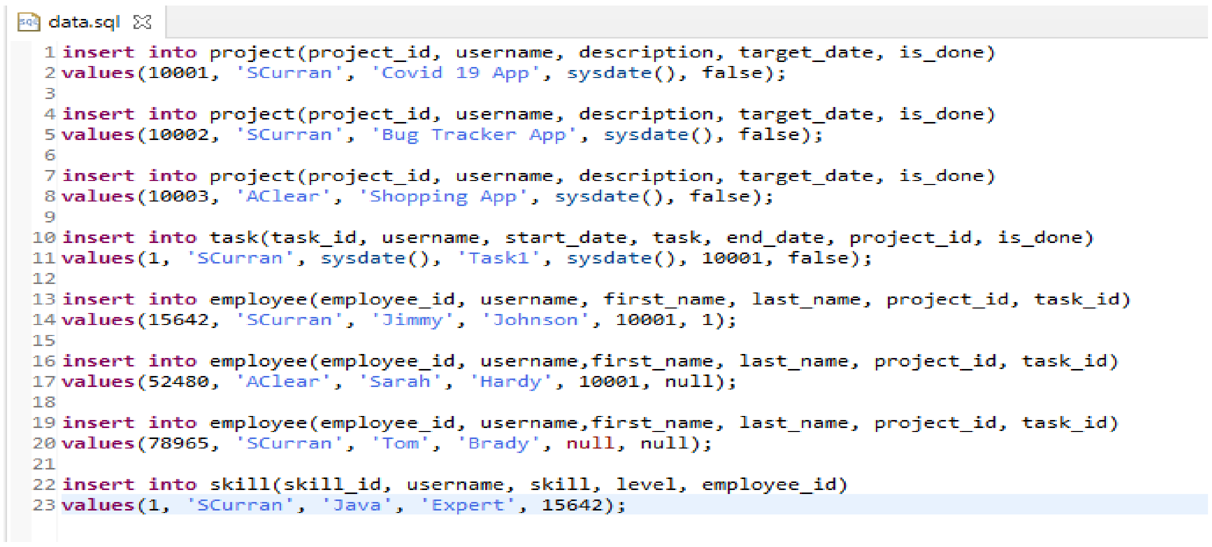
As a full stack application, the system makes a connection between the ReactJS frontend and the Java Spring backend. The backend uses the default localhost:8080 while the frontend was edited to localhost:4200 in order to avoid clashes when making the connection. The H2 database due to being built in to the Java Spring backend can be accessed from the localhost:8080/h2-console/ where the student can then connect to it as seen below.



Saved Settings: Generic H2 (Embedded)
Setting Name: Generic H2 (Embedded)
Driver Class: org.h2.Driver
JDBC URL: jdbc:h2:mem:testdb
User Name: sa
Password:

Figure 9: Connect to H2

Any dummy data can be inserted into the H2 database via the data.sql file seen below in the backend.



```sql
insert into project(project_id, username, description, target_date, is_done)
values(10001, 'SCurran', 'Covid 19 App', sysdate(), false);

insert into project(project_id, username, description, target_date, is_done)
values(10002, 'SCurran', 'Bug Tracker App', sysdate(), false);

insert into project(project_id, username, description, target_date, is_done)
values(10003, 'AClear', 'Shopping App', sysdate(), false);

insert into task(task_id, username, start_date, task, end_date, project_id, is_done)
values(1, 'SCurran', sysdate(), 'Task1', sysdate(), 10001, false);

insert into employee(employee_id, username, first_name, last_name, project_id, task_id)
values(15642, 'SCurran', 'Jimmy', 'Johnson', 10001, 1);

insert into employee(employee_id, username,first_name, last_name, project_id, task_id)
values(52480, 'AClear', 'Sarah', 'Hardy', 10001, null);

insert into employee(employee_id, username,first_name, last_name, project_id, task_id)
values(78965, 'SCurran', 'Tom', 'Brady', null, null);

insert into skill(skill_id, username, skill, level, employee_id)
values(1, 'SCurran', 'Java', 'Expert', 15642);
```

Figure 10: data.sql

Below is the visual of the architecture of the system:

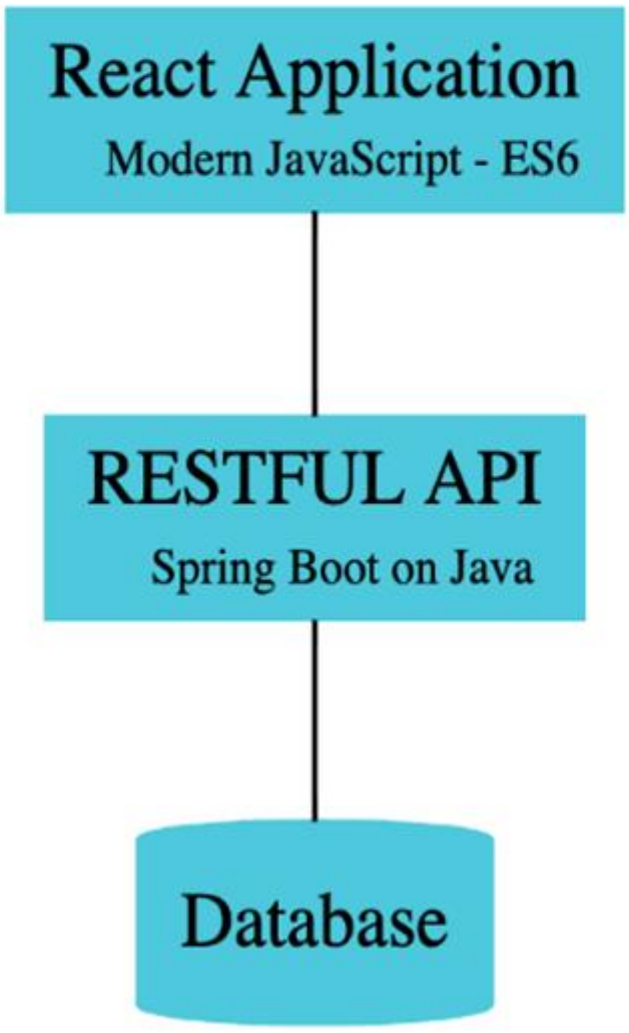


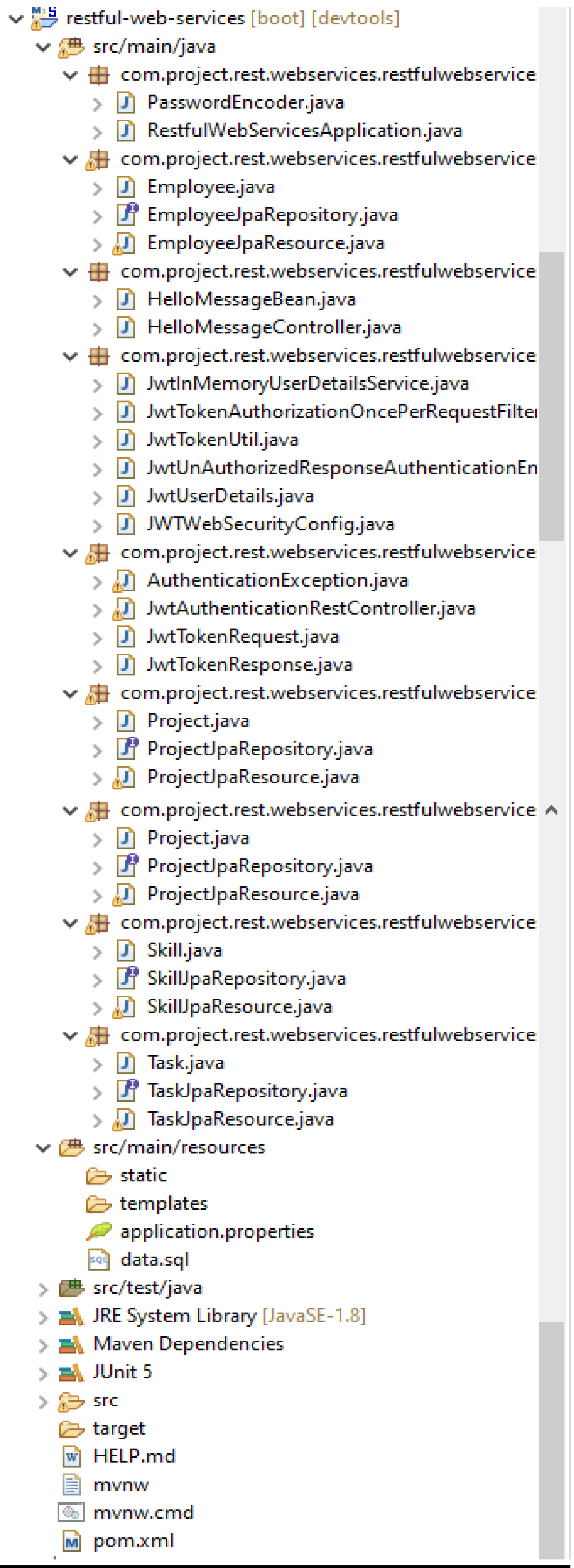
Figure 11: Visual of architecture

As seen above, the RESTful API is initialized using Spring boot. React then presents the User Interface. The database is hardcoded in eclipse.

5.1 File Breakdown

The following are the file breakdowns for both the backend in Eclipse and frontend in Visual Studio Code. I will go over some of the most important classes and jsx files and what their relevance is to the project.

5.1.1 Backend



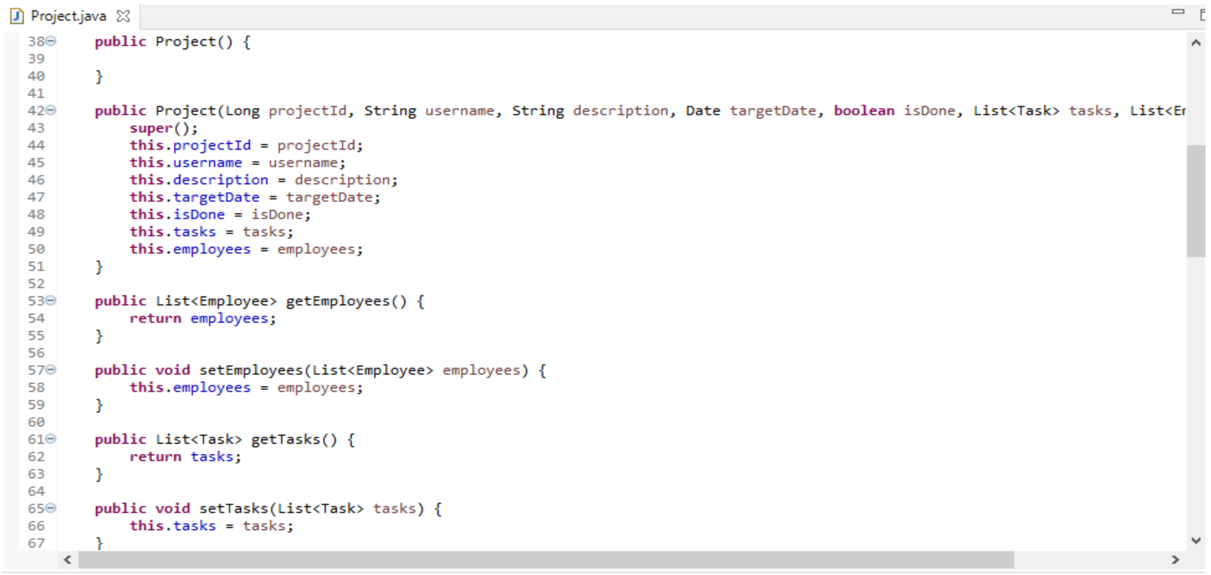
```
restful-web-services [boot] [devtools]
  src/main/java
    com.project.rest.webservices.restfulwebservice
      PasswordEncoder.java
      RestfulWebServicesApplication.java
    com.project.rest.webservices.restfulwebservice
      Employee.java
      EmployeeJpaRepository.java
      EmployeeJpaResource.java
    com.project.rest.webservices.restfulwebservice
      HelloMessageBean.java
      HelloMessageController.java
    com.project.rest.webservices.restfulwebservice
      JwtInMemoryUserDetailsService.java
      JwtTokenAuthorizationOncePerRequestFilte
      JwtTokenUtil.java
      JwtUnAuthorizedResponseAuthenticationEn
      JwtUserDetails.java
      JWTWebSecurityConfig.java
    com.project.rest.webservices.restfulwebservice
      AuthenticationException.java
      JwtAuthenticationRestController.java
      JwtTokenRequest.java
      JwtTokenResponse.java
    com.project.rest.webservices.restfulwebservice
      Project.java
      ProjectJpaRepository.java
      ProjectJpaResource.java
    com.project.rest.webservices.restfulwebservice
      Project.java
      ProjectJpaRepository.java
      ProjectJpaResource.java
    com.project.rest.webservices.restfulwebservice
      Skill.java
      SkillJpaRepository.java
      SkillJpaResource.java
    com.project.rest.webservices.restfulwebservice
      Task.java
      TaskJpaRepository.java
      TaskJpaResource.java
  src/main/resources
    static
    templates
    application.properties
    data.sql
  src/test/java
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  JUnit 5
  src
  target
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
```

Figure 12: Eclipse files

A common theme you have probably noticed from the breakdown above is the use of three particular files in package. The standard class files for Project, Task, Employee and Skill with a JpaRepository and JpaResource. But what are their purposes? For these explanations I have will be using the files from the Project package.

Project.Java

Project.Java acts as a JPA entity meaning what features in this class features in the actual Project table in the database.

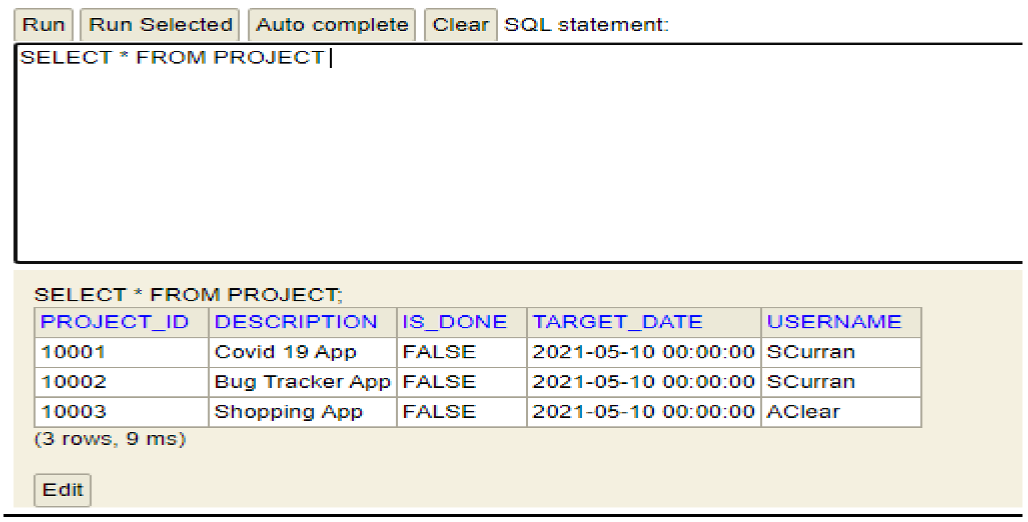


Project.java

```java
    public Project() {

    }

    public Project(Long projectId, String username, String description, Date targetDate, boolean isDone, List<Task> tasks, List<E
        super();
        this.projectId = projectId;
        this.username = username;
        this.description = description;
        this.targetDate = targetDate;
        this.isDone = isDone;
        this.tasks = tasks;
        this.employees = employees;
    }

    public List<Employee> getEmployees() {
        return employees;
    }

    public void setEmployees(List<Employee> employees) {
        this.employees = employees;
    }

    public List<Task> getTasks() {
        return tasks;
    }

    public void setTasks(List<Task> tasks) {
        this.tasks = tasks;
    }
```

Figure 13: Project.java

As you can see, it mostly features getters and setters as well as the Lists for the other object classes it links to where the project ID is the foreign key. Below is how it actually translates to the database.



Run | Run Selected | Auto complete | Clear | SQL statement:

```sql
SELECT * FROM PROJECT
```

SELECT * FROM PROJECT;

PROJECT_ID	DESCRIPTION	IS_DONE	TARGET_DATE	USERNAME
10001	Covid 19 App	FALSE	2021-05-10 00:00:00	SCurran
10002	Bug Tracker App	FALSE	2021-05-10 00:00:00	SCurran
10003	Shopping App	FALSE	2021-05-10 00:00:00	AClear

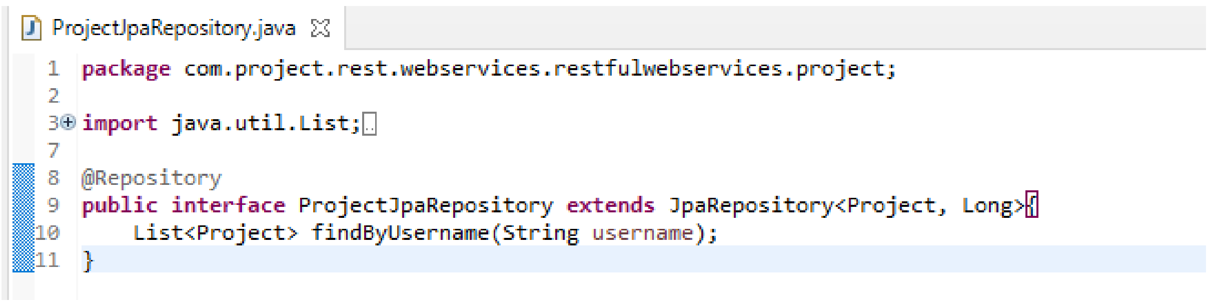
(3 rows, 9 ms)

Edit

Figure 14: Table in database

ProjectJpaRepository.Java

The ProjectJpaRepository is exactly as the title suggests, the repository for the Projects JPA resource. It is what allows the Project Resource to find specific usernames, id's or display all data when doing Get, Post, Put or Delete mapping.

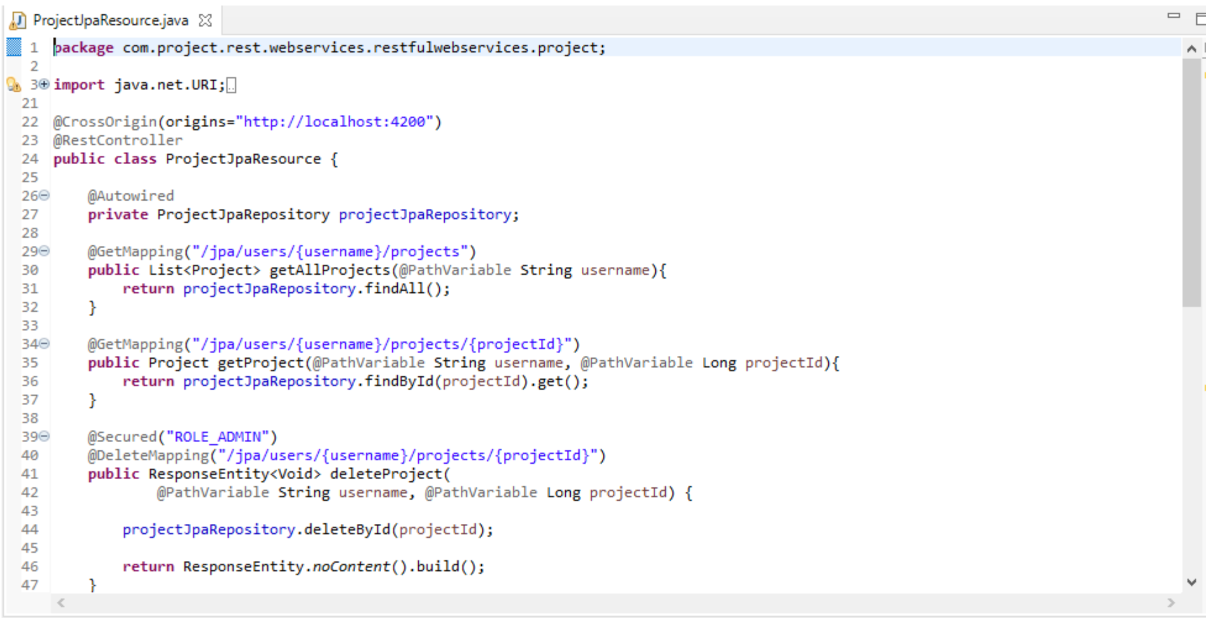


```java
package com.project.rest.webservices.restfulwebservices.project;

import java.util.List;

@Repository
public interface ProjectJpaRepository extends JpaRepository<Project, Long>{
    List<Project> findByUsername(String username);
}
```

Figure 15: ProjectJpaRepository.java

ProjectJpaResource.Java

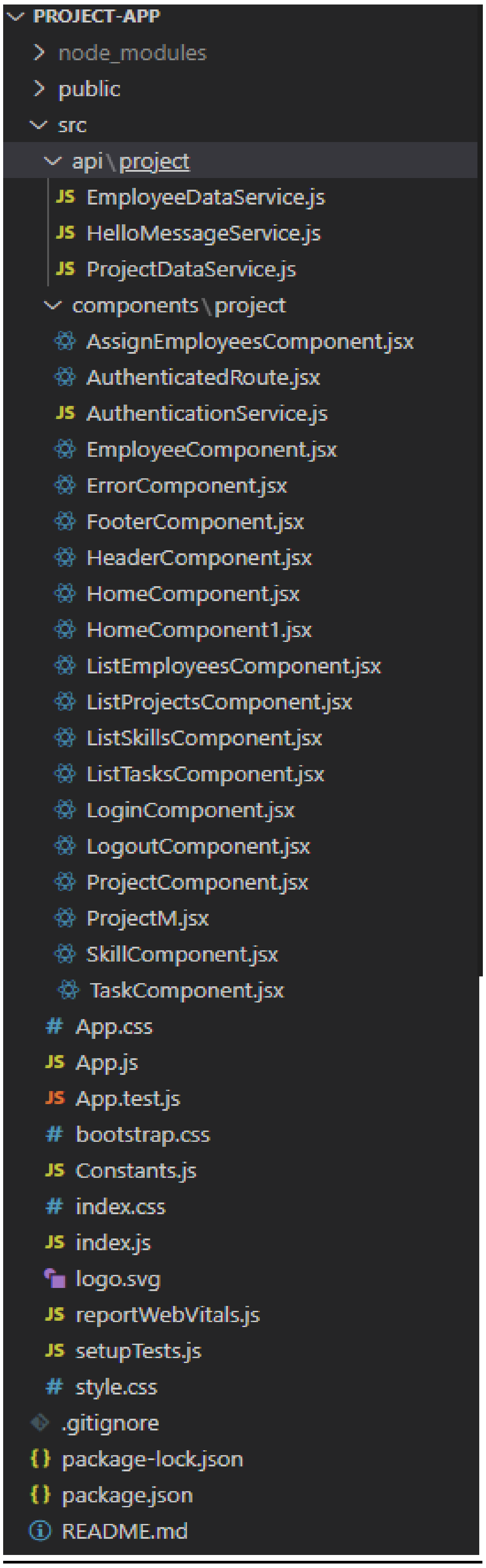
The ProjectJpaResource acts as the controller for the REST portion of the projects. This is where the mappings are done for our Gets, Posts, Puts or Deletes. The @RestController annotation is what allows the creation of our RESTful web services. The @Autowired annotation allows more control over where autowiring should be accomplished. The rest of the class is made of up or mappings with @Secured determining what is restricted to admins.



```java
package com.project.rest.webservices.restfulwebservices.project;

import java.net.URI;

@CrossOrigin(origins="http://localhost:4200")
@RestController
public class ProjectJpaResource {

    @Autowired
    private ProjectJpaRepository projectJpaRepository;

    @GetMapping("/jpa/users/{username}/projects")
    public List<Project> getAllProjects(@PathVariable String username){
        return projectJpaRepository.findAll();
    }

    @GetMapping("/jpa/users/{username}/projects/{projectId}")
    public Project getProject(@PathVariable String username, @PathVariable Long projectId){
        return projectJpaRepository.findById(projectId).get();
    }

    @Secured("ROLE_ADMIN")
    @DeleteMapping("/jpa/users/{username}/projects/{projectId}")
    public ResponseEntity<Void> deleteProject(
            @PathVariable String username, @PathVariable Long projectId) {

        projectJpaRepository.deleteById(projectId);

        return ResponseEntity.noContent().build();
    }
```

Figure 16: ProjectJpaResource.java

5.1.2 Frontend



```
PROJECT-APP
  node_modules
  public
  src
    api\project
      EmployeeDataService.js
      HelloMessageService.js
      ProjectDataService.js
    components\project
      AssignEmployeesComponent.jsx
      AuthenticatedRoute.jsx
      AuthenticationService.js
      EmployeeComponent.jsx
      ErrorComponent.jsx
      FooterComponent.jsx
      HeaderComponent.jsx
      HomeComponent.jsx
      HomeComponent1.jsx
      ListEmployeesComponent.jsx
      ListProjectsComponent.jsx
      ListSkillsComponent.jsx
      ListTasksComponent.jsx
      LoginComponent.jsx
      LogoutComponent.jsx
      ProjectComponent.jsx
      ProjectM.jsx
      SkillComponent.jsx
      TaskComponent.jsx
    App.css
    App.js
    App.test.js
    bootstrap.css
    Constants.js
    index.css
    index.js
    logo.svg
    reportWebVitals.js
    setupTests.js
    style.css
  .gitignore
  package-lock.json
  package.json
  README.md
```

Figure 17: Visual Studio Code files

For those reading with no experience of the REACT framework, the first word that is probably standing out to you right now is “component”.

What are components?

Components in REACT are simply the building blocks of any REACT app or in the students case, just the front end. There is no actual need to name any of these .jsx files “component” however. It was simply the naming convention the student chose.

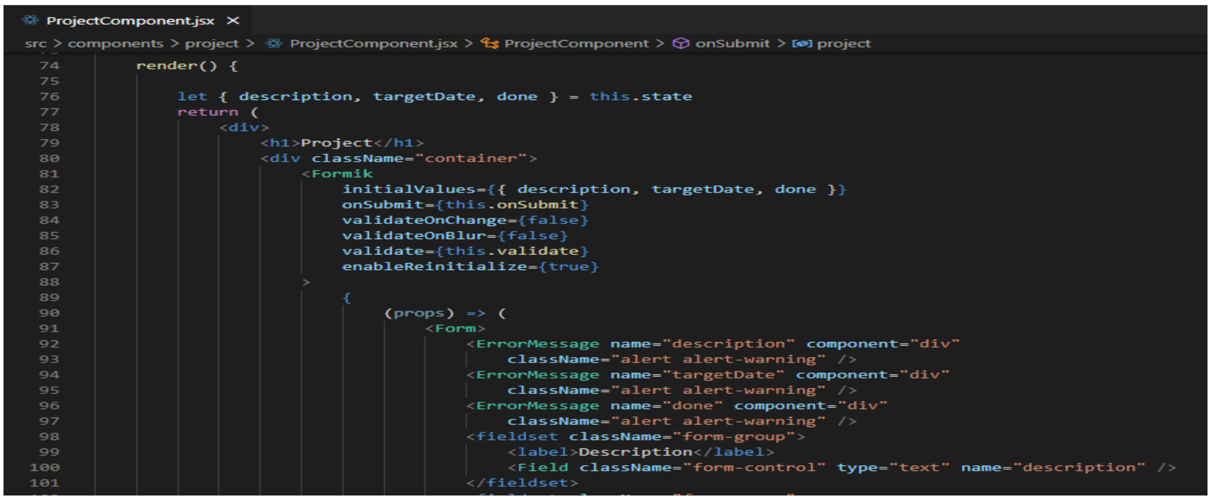
What are .jsx files?

JSX is a syntax extension of javascript. JSX is not actually 100% necessary in REACT but it makes development of UI far easier because it works as both a visual aid and allows REACT to show useful error and warning messages whenever there was a problem. This was particular useful in the students experience.

You will notice that there are .jsx files for each of our main tables from the backend; Project, Task, Employee and Skill. There are also List files for all of these. How these actually connect to our backend will be elaborated on in the next section. For the breakdown of the files, I will use the Project versions as the example again.

ProjectComponent.jsx

The project component is for both the creation and updating on an individual project. As you can see, it will render out of all the specific columns of the new or current project and data can be entered into it which will be saved.

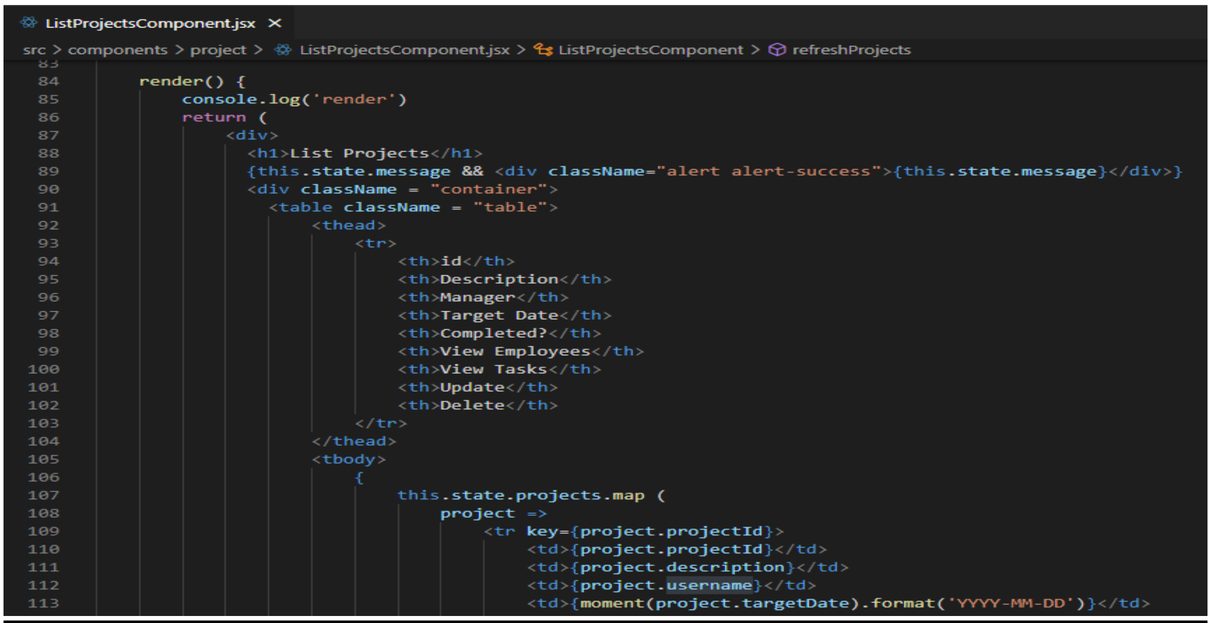

ProjectComponent.jsx
src > components > project > ProjectComponent.jsx > ProjectComponent > onSubmit > project

```
    render() {

        let { description, targetDate, done } = this.state
        return (
            <div>
                <h1>Project</h1>
                <div className="container">
                    <Formik
                        initialValues={{ description, targetDate, done }}
                        onSubmit={this.onSubmit}
                        validateOnChange={false}
                        validateOnBlur={false}
                        validate={this.validate}
                        enableReinitialize={true}
                    >
                        {
                            (props) => (
                                <Form>
                                    <ErrorMessage name="description" component="div"
                                        className="alert alert-warning" />
                                    <ErrorMessage name="targetDate" component="div"
                                        className="alert alert-warning" />
                                    <ErrorMessage name="done" component="div"
                                        className="alert alert-warning" />
                                    <fieldset className="form-group">
                                        <label>Description</label>
                                        <Field className="form-control" type="text" name="description" />
                                    </fieldset>
```

Figure 18: ProjectComponent.jsx

ListProjectsComponent.jsx

The List Projects Component is how the all of the projects actually display with their info along with the View Employees and View Tasks buttons. The option to add a new Project, update an existing one and delete an existing are also added as buttons.

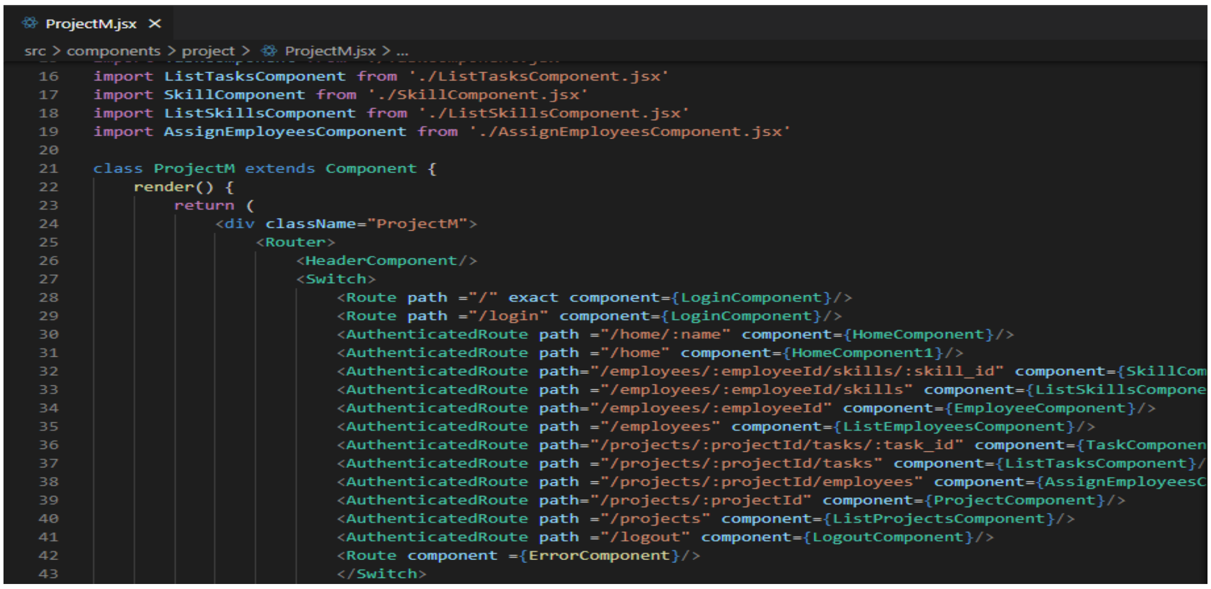

ListProjectsComponent.jsx ✕
src > components > project > ListProjectsComponent.jsx > ListProjectsComponent > refreshProjects

```
    render() {
        console.log('render')
        return (
            <div>
                <h1>List Projects</h1>
                {this.state.message && <div className="alert alert-success">{this.state.message}</div>}
                <div className = "container">
                  <table className = "table">
                      <thead>
                          <tr>
                              <th>id</th>
                              <th>Description</th>
                              <th>Manager</th>
                              <th>Target Date</th>
                              <th>Completed?</th>
                              <th>View Employees</th>
                              <th>View Tasks</th>
                              <th>Update</th>
                              <th>Delete</th>
                          </tr>
                      </thead>
                      <tbody>
                          {
                              this.state.projects.map (
                                  project =>
                                      <tr key={project.projectId}>
                                          <td>{project.projectId}</td>
                                          <td>{project.description}</td>
                                          <td>{project.username}</td>
                                          <td>{moment(project.targetDate).format('YYYY-MM-DD')}</td>
```

Figure 19: ListProjectsComponent.jsx

ProjectM.jsx

It should be noted that nothing from any of these .jsx files would actually display without the ProjectM.jsx which is the main file for the front end of the app. As you can see, it imports all the other files and determines the paths.

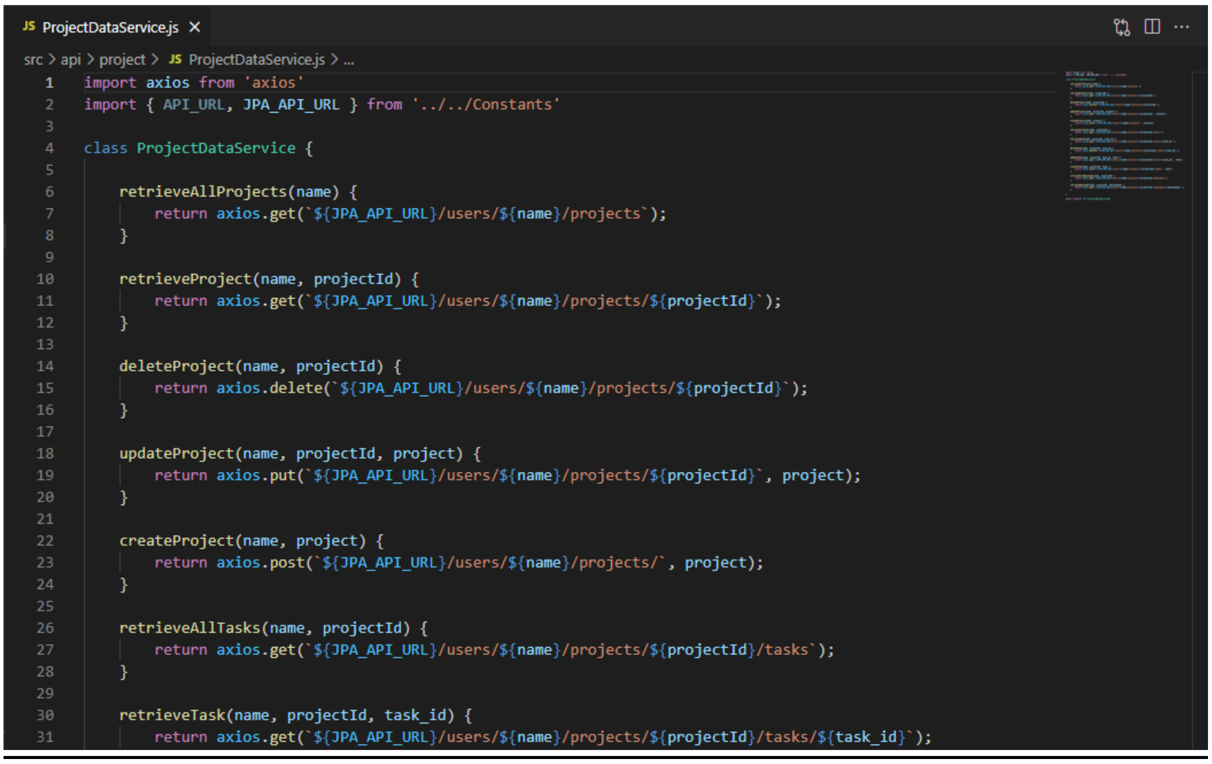

ProjectM.jsx ✕
src > components > project > ProjectM.jsx > ...

```
import ListTasksComponent from './ListTasksComponent.jsx'
import SkillComponent from './SkillComponent.jsx'
import ListSkillsComponent from './ListSkillsComponent.jsx'
import AssignEmployeesComponent from './AssignEmployeesComponent.jsx'

class ProjectM extends Component {
    render() {
        return (
            <div className="ProjectM">
                <Router>
                    <HeaderComponent/>
                    <Switch>
                        <Route path ="/" exact component={LoginComponent}/>
                        <Route path ="/login" component={LoginComponent}/>
                        <AuthenticatedRoute path ="/home/:name" component={HomeComponent}/>
                        <AuthenticatedRoute path ="/home" component={HomeComponent1}/>
                        <AuthenticatedRoute path="/employees/:employeeId/skills/:skill_id" component={SkillCom
                        <AuthenticatedRoute path ="/employees/:employeeId/skills" component={ListSkillsCompone
                        <AuthenticatedRoute path="/employees/:employeeId" component={EmployeeComponent}/>
                        <AuthenticatedRoute path ="/employees" component={ListEmployeesComponent}/>
                        <AuthenticatedRoute path="/projects/:projectId/tasks/:task_id" component={TaskComponen
                        <AuthenticatedRoute path ="/projects/:projectId/tasks" component={ListTasksComponent}/
                        <AuthenticatedRoute path ="/projects/:projectId/employees" component={AssignEmployeesC
                        <AuthenticatedRoute path="/projects/:projectId" component={ProjectComponent}/>
                        <AuthenticatedRoute path ="/projects" component={ListProjectsComponent}/>
                        <AuthenticatedRoute path ="/logout" component={LogoutComponent}/>
                        <Route component ={ErrorComponent}/>
                        </Switch>
```

Figure 20: ProjectM.jsx

5.2 How does the frontend and backend connect?

The ProjectDataService.js acts as the connection point between the backend and frontend. All of the mapping features which were shown in the JPA Resource come through here and all of these are referenced in the Project Component and List Projects Component.



```javascript
import axios from 'axios'
import { API_URL, JPA_API_URL } from '../../Constants'

class ProjectDataService {

    retrieveAllProjects(name) {
        return axios.get(`${JPA_API_URL}/users/${name}/projects`);
    }

    retrieveProject(name, projectId) {
        return axios.get(`${JPA_API_URL}/users/${name}/projects/${projectId}`);
    }

    deleteProject(name, projectId) {
        return axios.delete(`${JPA_API_URL}/users/${name}/projects/${projectId}`);
    }

    updateProject(name, projectId, project) {
        return axios.put(`${JPA_API_URL}/users/${name}/projects/${projectId}`, project);
    }

    createProject(name, project) {
        return axios.post(`${JPA_API_URL}/users/${name}/projects/`, project);
    }

    retrieveAllTasks(name, projectId) {
        return axios.get(`${JPA_API_URL}/users/${name}/projects/${projectId}/tasks`);
    }

    retrieveTask(name, projectId, task_id) {
        return axios.get(`${JPA_API_URL}/users/${name}/projects/${projectId}/tasks/${task_id}`);
```

Figure 21: ProjectDataService.js

5.3 Language breakdown

If github is to be believed, then the app is approximately 54.9% Javascript and 42.4% Java. This of course makes sense considering these are the two primary languages of the frontend and backend. There are minor instances of HTML and CSS in the frontend portion. The CSS of course being the bootstrap used.

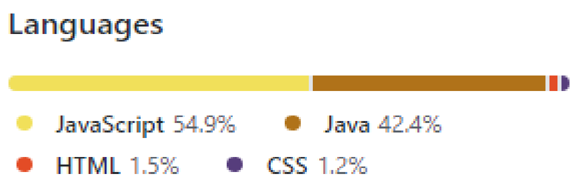
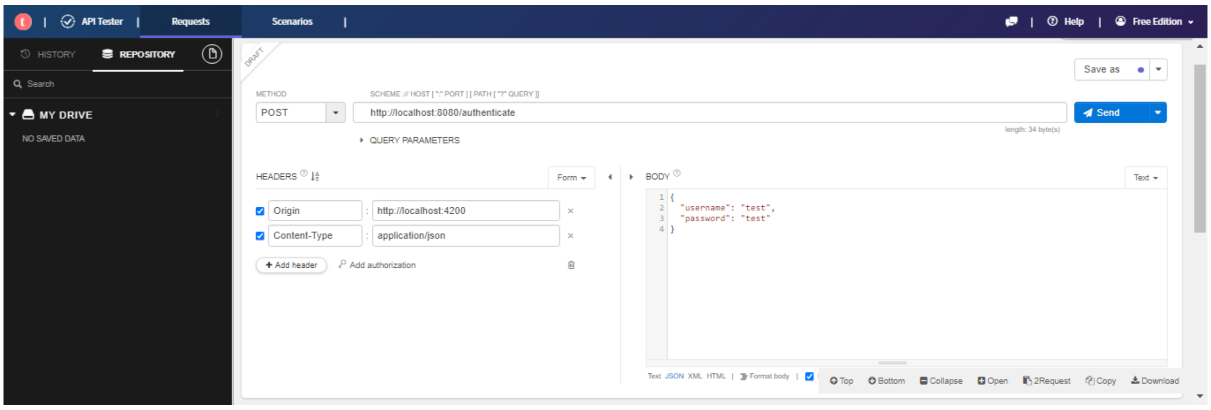


Figure 22: Github language breakdown

6. Tests

6.1 API Testing

API (Application Programming Interface) Testing was done throughout the development of the app. API is a type of automation testing. Its purpose throughout the app was the make sure the Get, Post, Put and Delete features were all working as well as the security. The type of API tester that was used was Talend API Tester which is an extension that can be added to google chrome. For example in the image below, the student tested with authentication of the app with a username and password.



METHOD: POST
SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]: http://localhost:8080/authenticate
length: 34 byte(s)

HEADERS

Origin	http://localhost:4200
Content-Type	application/json

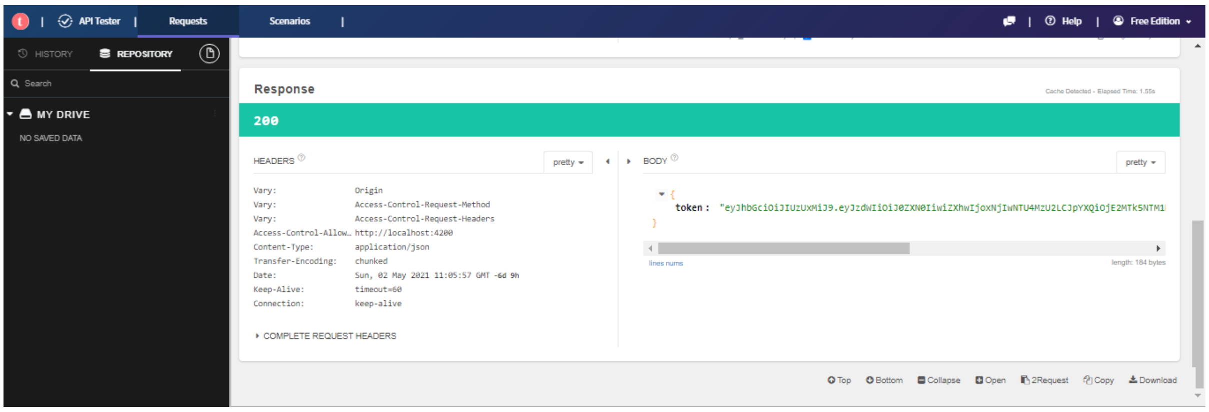
BODY

```
{
  "username": "test",
  "password": "test"
}
```

Text JSON XML HTML | Format body

Figure 23: Talend API Tester part 1

The student then received a HTTP 200 response. This confirms that the request has succeeded. The student can then move on to developing and testing the next feature. On the other hand, if the student had got for example a HTTP 401 response, then this would be an error with the authentication and the student would need to figure out the problem.



Response
Cache Detected - Elapsed Time: 1.55s

200

HEADERS

```
Vary:                  Origin
Vary:                  Access-Control-Request-Method
Vary:                  Access-Control-Request-Headers
Access-Control-Allow… http://localhost:4200
Content-Type:          application/json
Transfer-Encoding:     chunked
Date:                  Sun, 02 May 2021 11:05:57 GMT -6d 9h
Keep-Alive:            timeout=60
Connection:            keep-alive
```

COMPLETE REQUEST HEADERS

BODY

```
{
  token : "eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJ0ZXN0IiwiZXhwIjoxNjIwNTU4MzU2LCJpYXQiOjE2MTk5NTM1
}
```

length: 184 bytes

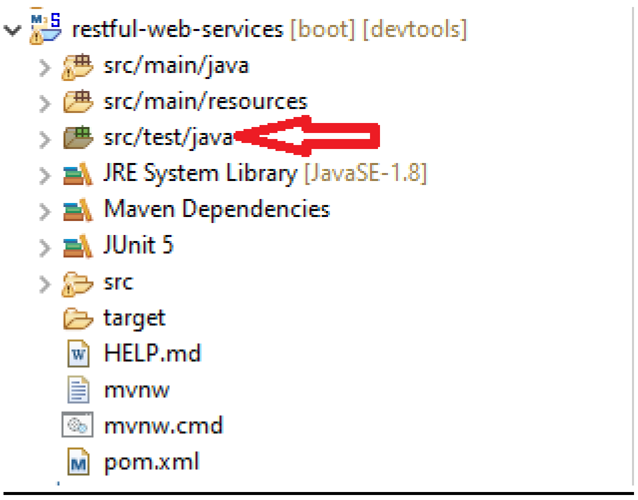
Figure 24: Talend API Tester part 2

Why use Talend API Tester?

The API tester the student used throughout their course for assignments was Postman, however Talend API Tester is not only much quicker to access due to being a chrome extension that can be accessed with the click of a button but the student also found it overall more user friendly.

6.2 JUnit Testing

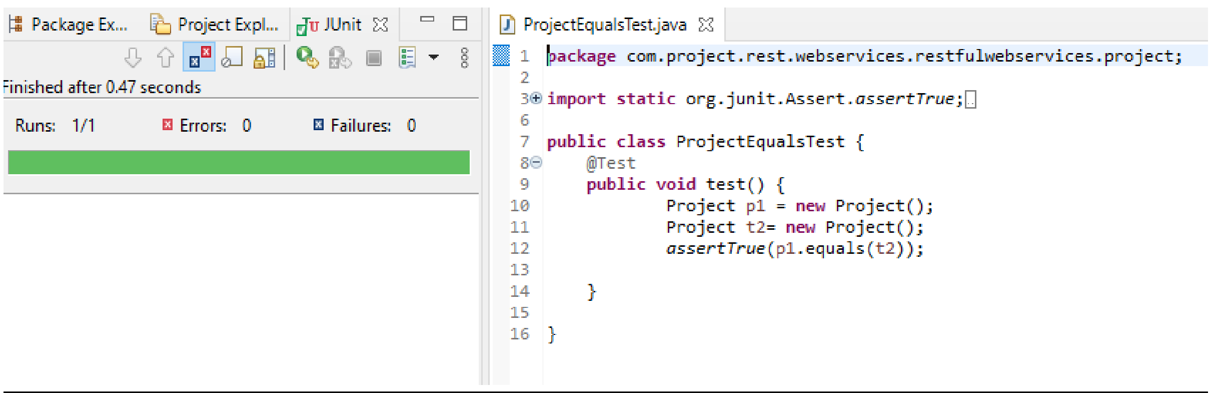
JUnit testing was done at certain points during development. JUnit testing is a form of white box testing. Java Spring has JUnit testing built into the pom.xml and the maven project download from Spring Initializr automatically came with a test suite as seen below.



restful-web-services [boot] [devtools]
- src/main/java
- src/main/resources
- src/test/java
- JRE System Library [JavaSE-1.8]
- Maven Dependencies
- JUnit 5
- src
- target
- HELP.md
- mvnw
- mvnw.cmd
- pom.xml

Figure 25: Test Suite

By running tests on snippets of code, the student could check to see if it’s working properly such as for example the test for the equals if statement in project.java below.



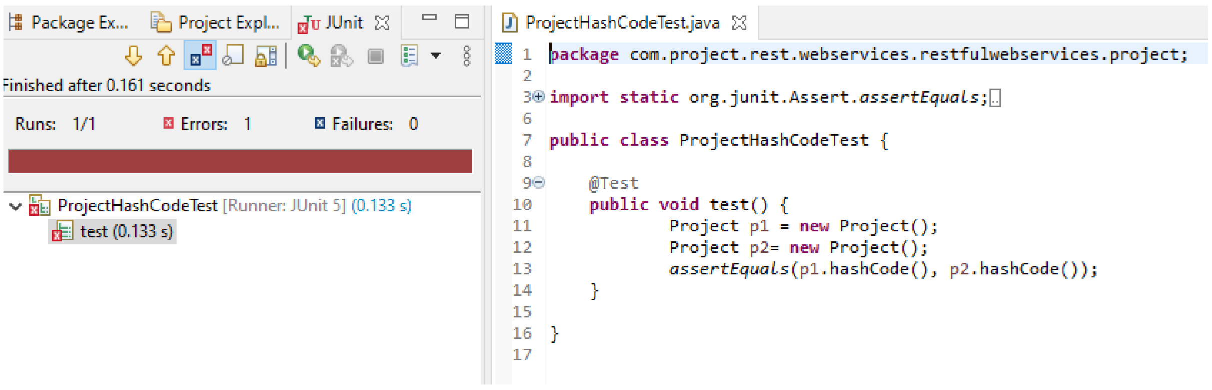
Finished after 0.47 seconds

Runs: 1/1 Errors: 0 Failures: 0

```
package com.project.rest.webservices.restfulwebservices.project;

import static org.junit.Assert.assertTrue;

public class ProjectEqualsTest {
    @Test
    public void test() {
            Project p1 = new Project();
            Project t2= new Project();
            assertTrue(p1.equals(t2));

    }

}
```

Figure 26: ProjectEqualsTest.java

Whenever a test failed such as the test on the hashcode() from project.java below, the student could go back and investigate where it is going wrong and how it could be fixed.



```
Finished after 0.161 seconds
Runs:  1/1      Errors:  1      Failures:  0
ProjectHashCodeTest [Runner: JUnit 5] (0.133 s)
    test (0.133 s)
```

ProjectHashCodeTest.java

```java
package com.project.rest.webservices.restfulwebservices.project;

import static org.junit.Assert.assertEquals;

public class ProjectHashCodeTest {

    @Test
    public void test() {
            Project p1 = new Project();
            Project p2= new Project();
            assertEquals(p1.hashCode(), p2.hashCode());
    }

}
```

Figure 27: ProjectHashCodeTest.java

Overall the JUnit tests were useful as they allowed the student to develop more readable, reliable and bug-free code which helped build confidence during development.

6.3 Black Box Testing

Black box tests were done to test out all of the main requirements and make sure they are working as expected. Most of the important tests have been included below. The type of black box testing done specifically was "functional testing" because all of the main functions were tested. For these tests, we will be using the username "SCurran" which has the admin role except for the last test where we will use one of the logins (username: "JJohnson") with the user role. Dummy data has also been entered into the tables from the data.sql file in the backend.

6.3.1 Test Cases

Test Case 1 – Testing Login with valid details.

Objective: To login to the app using established valid login details.

Inputs: username, password, login (button click)

Expected Output: Login will happen on click with no issue.

Test - Input:

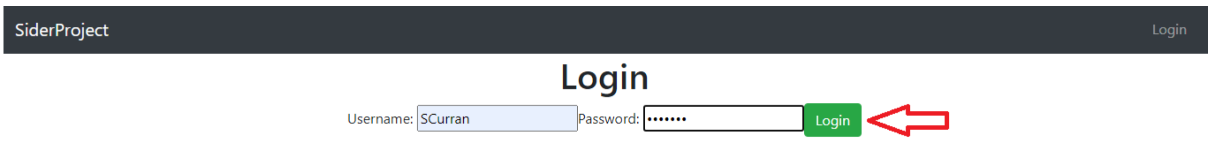
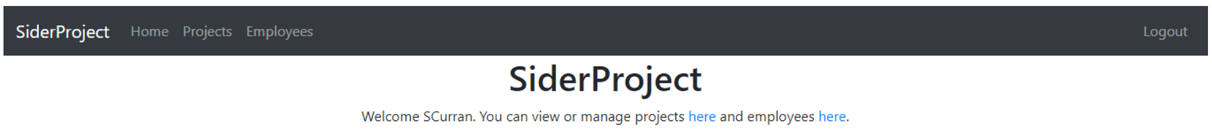


Figure 28

Test - Output:



Welcome SCurran. You can view or manage projects here and employees here.

Figure 29

Test - Result: **Pass**

The login works as intended.

Test Case 2 – Testing Login with invalid details.

Objective: To try to login to the app with invalid details and see if error message displays.

Inputs: username (invalid), password (invalid), login (button click)

Expected Output: Error message will display

Test - Input:

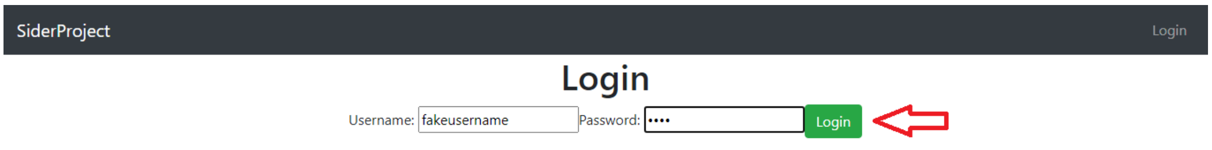
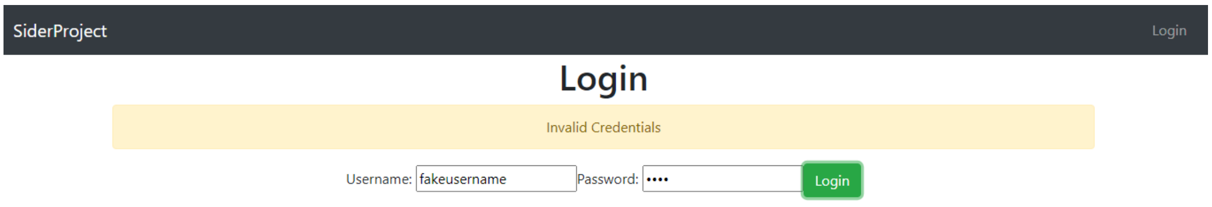


Figure 30

Test - Output:



Invalid Credentials

Figure 31

Test - Result: **Pass**

The error message displays for invalid details as intended.

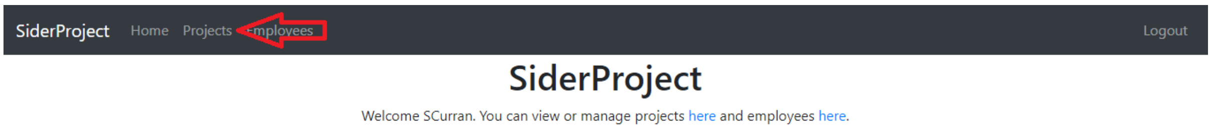
Test Case 3 – Fetching all projects from database.

Objective: To have all projects from database listed when going to project page

Inputs: Projects tab (click)

Expected Output: All current projects will display

Test - Input:

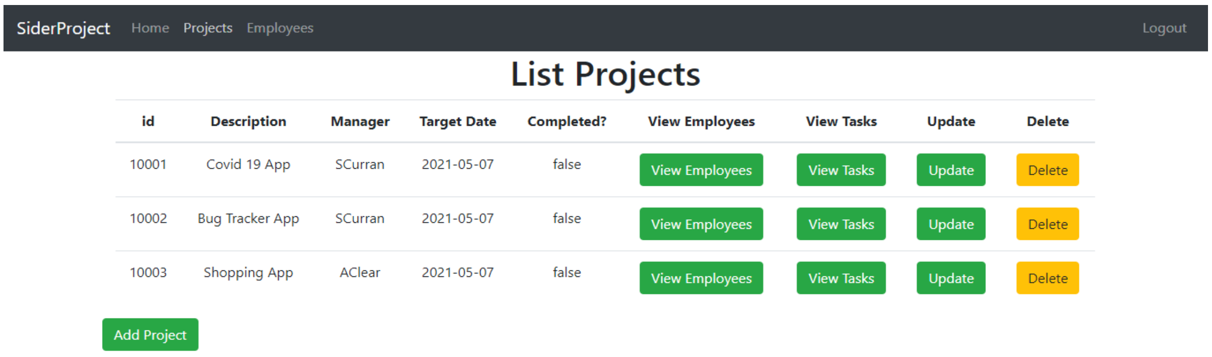

SiderProject Home Projects Employees Logout

SiderProject

Welcome SCurran. You can view or manage projects here and employees here.

Figure 32

Test - Output:


SiderProject Home Projects Employees Logout

List Projects

id	Description	Manager	Target Date	Completed?	View Employees	View Tasks	Update	Delete
10001	Covid 19 App	SCurran	2021-05-07	false	View Employees	View Tasks	Update	Delete
10002	Bug Tracker App	SCurran	2021-05-07	false	View Employees	View Tasks	Update	Delete
10003	Shopping App	AClear	2021-05-07	false	View Employees	View Tasks	Update	Delete

Add Project

Test - Result: Pass

All current projects display as intended.

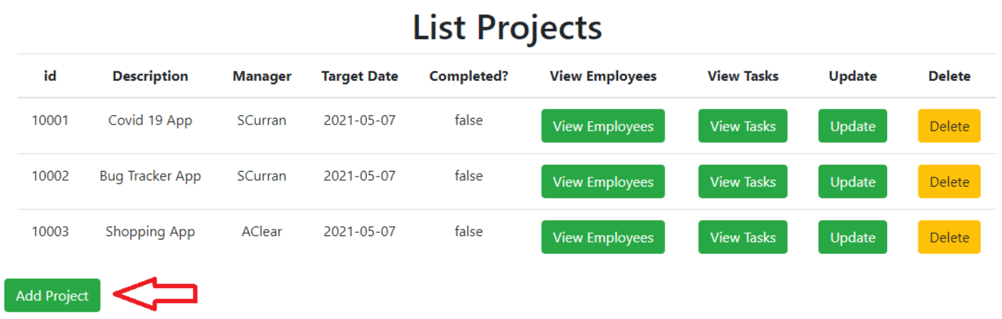
Test Case 4 – Add new project to list.

Objective: To add a new project to the projects list.

Inputs: Add Project (Button click), Description, Target Date, Completed? (True/False), Save (Button click)

Expected Output: New project will be added to list.

Test – Input 1:

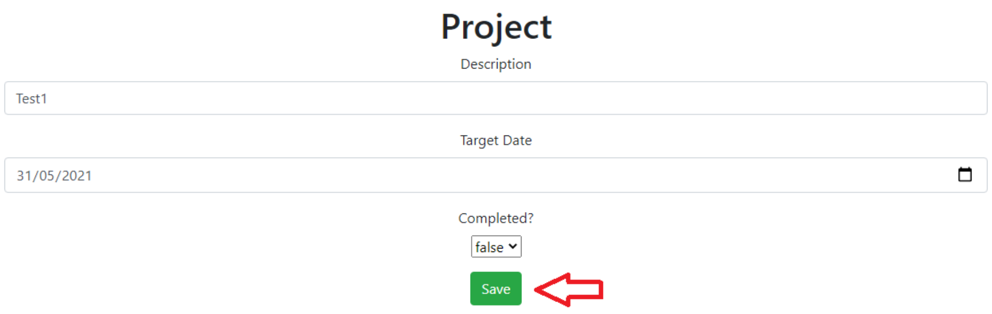

List Projects

id	Description	Manager	Target Date	Completed?	View Employees	View Tasks	Update	Delete
10001	Covid 19 App	SCurran	2021-05-07	false	View Employees	View Tasks	Update	Delete
10002	Bug Tracker App	SCurran	2021-05-07	false	View Employees	View Tasks	Update	Delete
10003	Shopping App	AClear	2021-05-07	false	View Employees	View Tasks	Update	Delete

Add Project

Figure 33

Test – Input 2:


Project

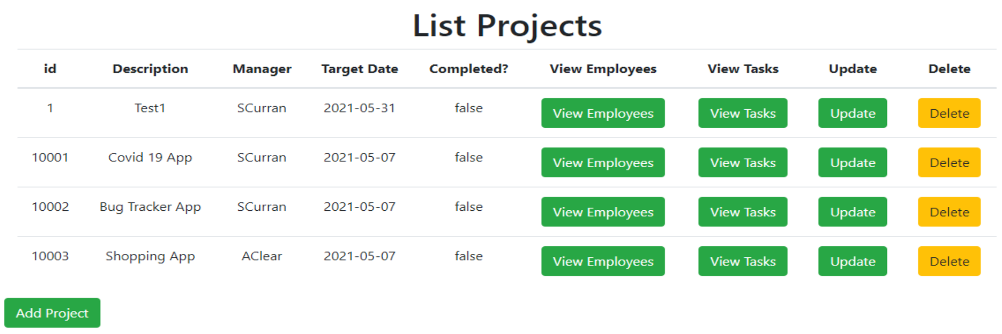
Description: Test1

Target Date: 31/05/2021

Completed? false

Figure 34

Test - Output:


List Projects

id	Description	Manager	Target Date	Completed?	View Employees	View Tasks	Update	Delete
1	Test1	SCurran	2021-05-31	false	View Employees	View Tasks	Update	Delete
10001	Covid 19 App	SCurran	2021-05-07	false	View Employees	View Tasks	Update	Delete
10002	Bug Tracker App	SCurran	2021-05-07	false	View Employees	View Tasks	Update	Delete
10003	Shopping App	AClear	2021-05-07	false	View Employees	View Tasks	Update	Delete

Figure 35

Test - Result: Pass

The project is added as intended.

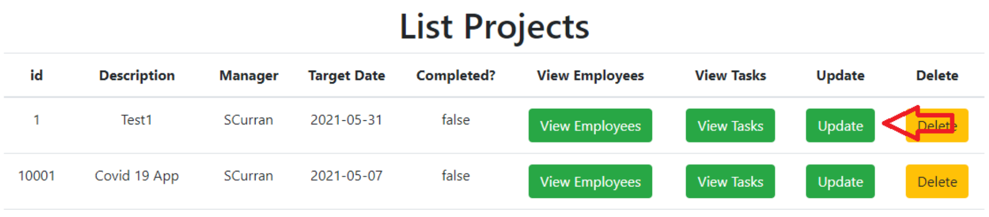
Test Case 5 – Update a current project.

Objective: To update a project from the projects list.

Inputs: Update (Button click), Description, Target Date, Completed? (True/False), Save (Button click)

Expected Output: Edited details will now be displayed instead.

Test – Input 1:

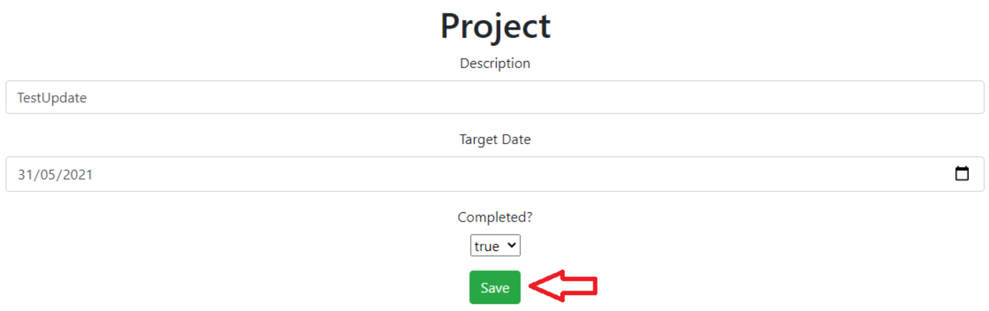


List Projects

id	Description	Manager	Target Date	Completed?	View Employees	View Tasks	Update	Delete
1	Test1	SCurran	2021-05-31	false	View Employees	View Tasks	Update	Delete
10001	Covid 19 App	SCurran	2021-05-07	false	View Employees	View Tasks	Update	Delete

Figure 36

Test – Input 2:



Project

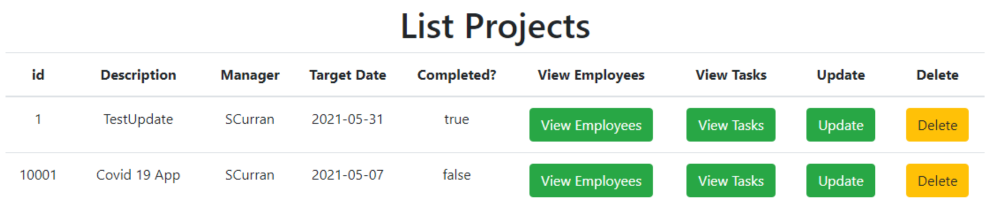
Description: TestUpdate

Target Date: 31/05/2021

Completed? true

Figure 37

Test - Output:



List Projects

id	Description	Manager	Target Date	Completed?	View Employees	View Tasks	Update	Delete
1	TestUpdate	SCurran	2021-05-31	true	View Employees	View Tasks	Update	Delete
10001	Covid 19 App	SCurran	2021-05-07	false	View Employees	View Tasks	Update	Delete

Figure 38

Test - Result: Pass

The project is updated as intended.

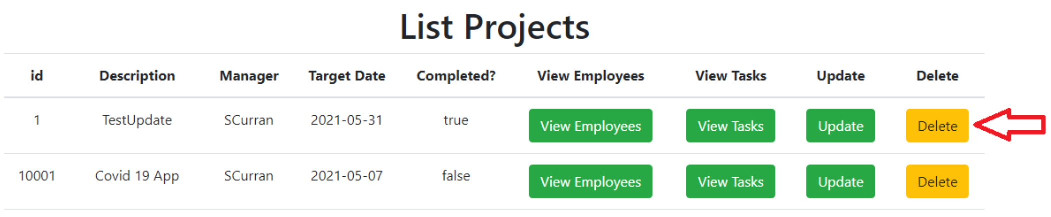
Test Case 6 – Delete a current project.

Objective: To delete a project from the projects list.

Inputs: Delete (Button click)

Expected Output: Project will be deleted from list.

Test – Input:

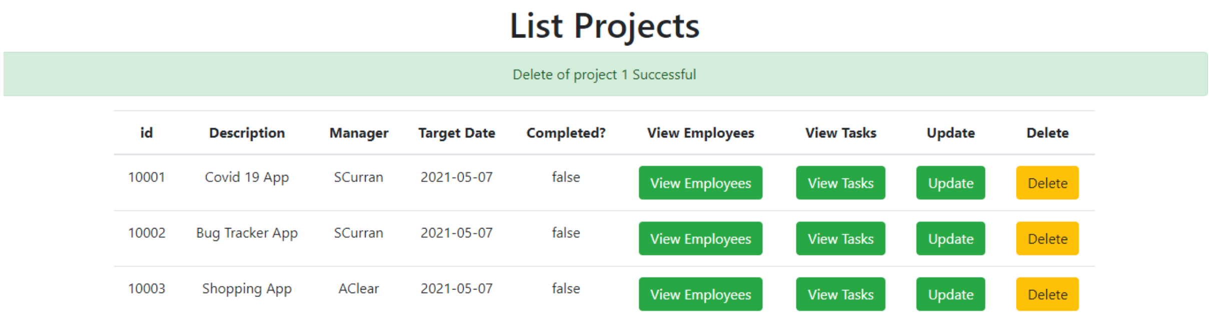


List Projects

id	Description	Manager	Target Date	Completed?	View Employees	View Tasks	Update	Delete
1	TestUpdate	SCurran	2021-05-31	true	View Employees	View Tasks	Update	Delete
10001	Covid 19 App	SCurran	2021-05-07	false	View Employees	View Tasks	Update	Delete

Figure 39

Test - Output:



List Projects

Delete of project 1 Successful

id	Description	Manager	Target Date	Completed?	View Employees	View Tasks	Update	Delete
10001	Covid 19 App	SCurran	2021-05-07	false	View Employees	View Tasks	Update	Delete
10002	Bug Tracker App	SCurran	2021-05-07	false	View Employees	View Tasks	Update	Delete
10003	Shopping App	AClear	2021-05-07	false	View Employees	View Tasks	Update	Delete

Figure 40

Test - Result: Pass

The project has been deleted as intended.

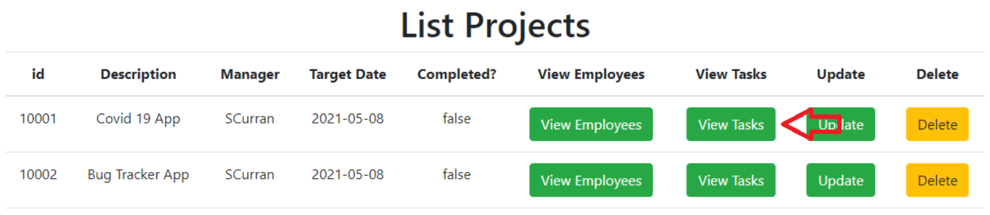
Test Case 7 – Add task to a specific project.

Objective: To add a task to a selected project from the list.

Inputs: View Tasks (Button click), Add Task (Button click), Start date, Task, End date, Completed? (True/False), Save (Button click)

Expected Output: Task will be added to specified project.

Test – Input 1:

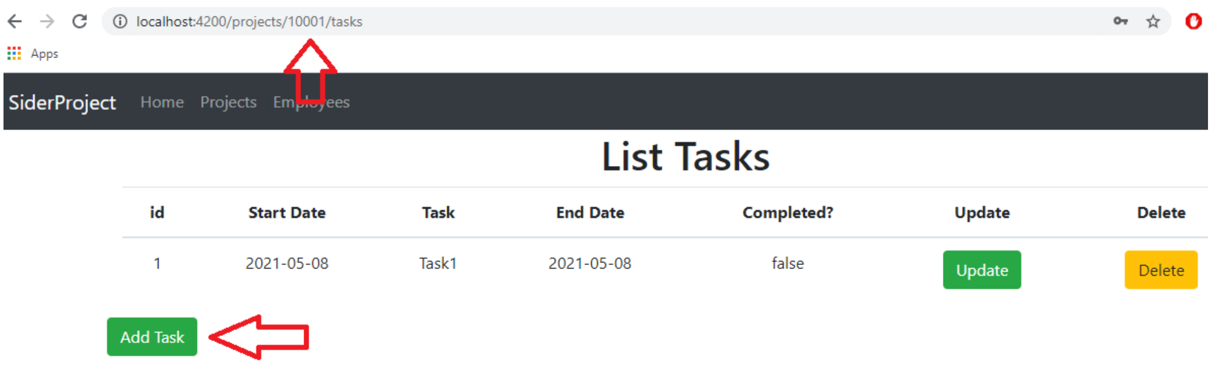


List Projects

id	Description	Manager	Target Date	Completed?	View Employees	View Tasks	Update	Delete
10001	Covid 19 App	SCurran	2021-05-08	false	View Employees	View Tasks	Update	Delete
10002	Bug Tracker App	SCurran	2021-05-08	false	View Employees	View Tasks	Update	Delete

Figure 41

Test – Input 2:



localhost:4200/projects/10001/tasks

SiderProject Home Projects Employees

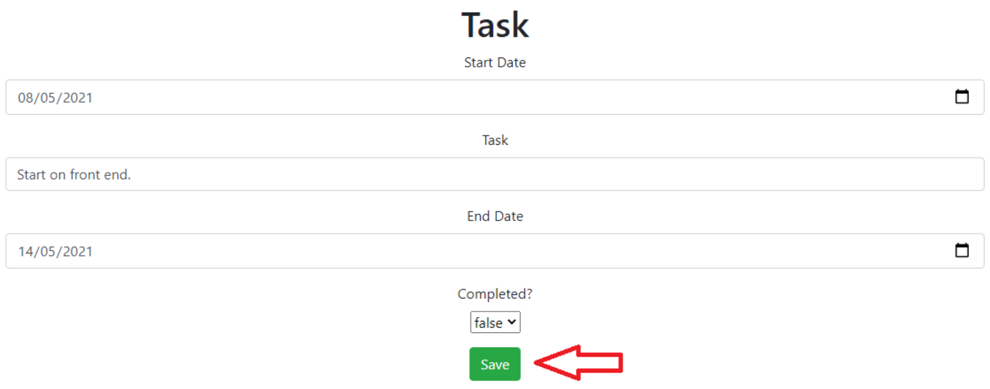
List Tasks

id	Start Date	Task	End Date	Completed?	Update	Delete
1	2021-05-08	Task1	2021-05-08	false	Update	Delete

Add Task

Figure 42

Test – Input 3:



Task

Start Date: 08/05/2021

Task: Start on front end.

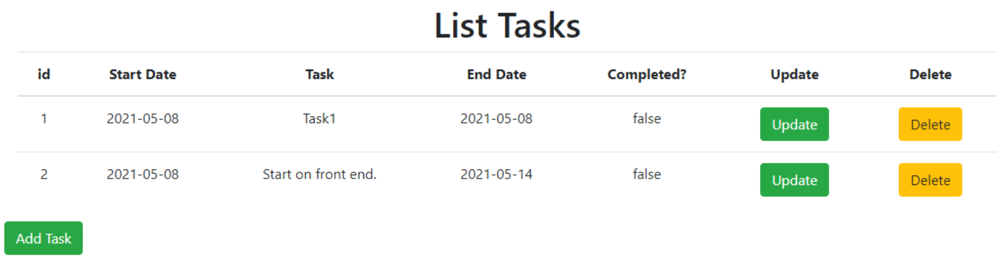
End Date: 14/05/2021

Completed?: false

Save

Figure 43

Test - Output:



List Tasks

id	Start Date	Task	End Date	Completed?	Update	Delete
1	2021-05-08	Task1	2021-05-08	false	Update	Delete
2	2021-05-08	Start on front end.	2021-05-14	false	Update	Delete

Add Task

Figure 44

Test - Result: Pass

The task has been added to the specific project as expected.

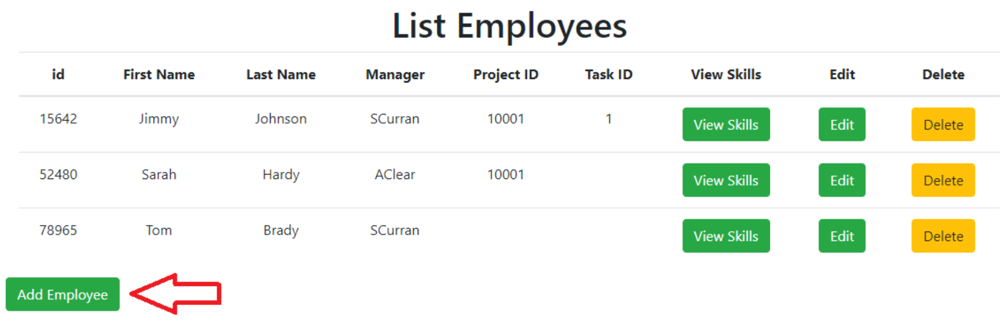
Test Case 8 – Add Employee to employees list and assign them to specific project.

Objective: To add a project to the employees list and assign a project.

Inputs: Add Employee (Button click), First Name, Last Name, Project ID, Task ID, Save (Button click)

Expected Output: Employee will added to list and assigned to a project with no issues.

Test – Input 1:

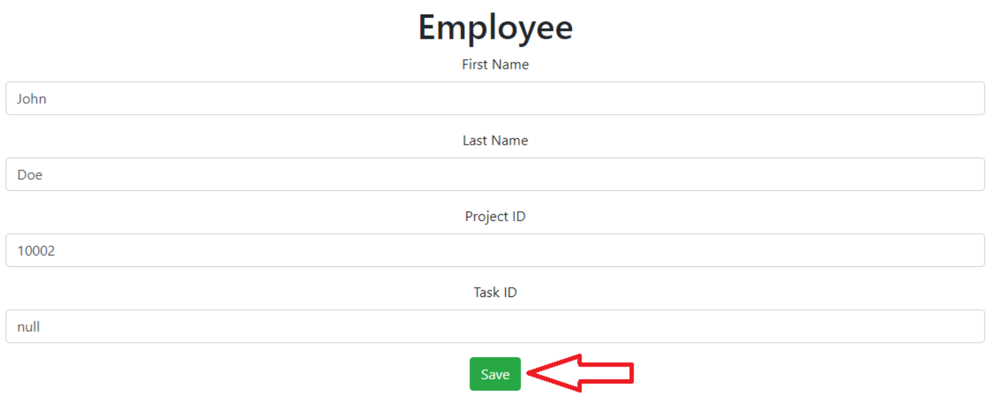


List Employees

id	First Name	Last Name	Manager	Project ID	Task ID	View Skills	Edit	Delete
15642	Jimmy	Johnson	SCurran	10001	1	View Skills	Edit	Delete
52480	Sarah	Hardy	AClear	10001		View Skills	Edit	Delete
78965	Tom	Brady	SCurran			View Skills	Edit	Delete

Figure 45

Test – Input 2:



Employee

First Name: John

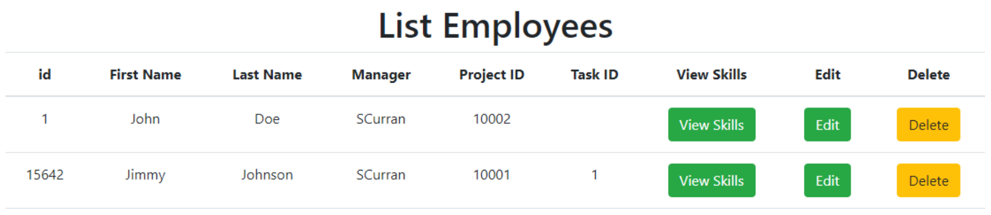
Last Name: Doe

Project ID: 10002

Task ID: null

Figure 46

Test - Output:



List Employees

id	First Name	Last Name	Manager	Project ID	Task ID	View Skills	Edit	Delete
1	John	Doe	SCurran	10002		View Skills	Edit	Delete
15642	Jimmy	Johnson	SCurran	10001	1	View Skills	Edit	Delete

Figure 47

Test - Result: Pass

The employee has been added to the employees list with the relevant project ID as expected.

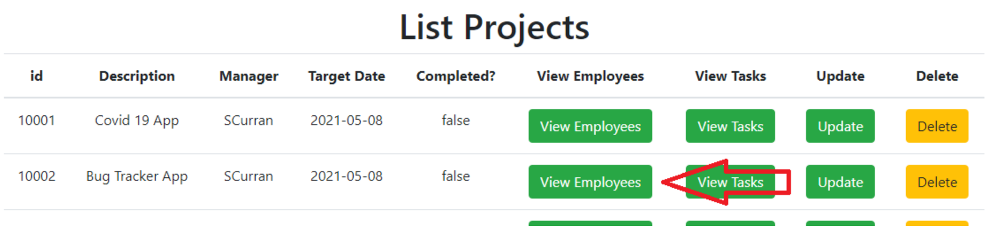
Test Case 9 – Check that employees for specified project are displaying.

Objective: To make sure the employee created in Test Case 7 above is now showing in the project it was assigned too.

Inputs: View Employees (Button click)

Expected Output: Employee assigned to project will be showing and no others from employee database.

Test – Input:

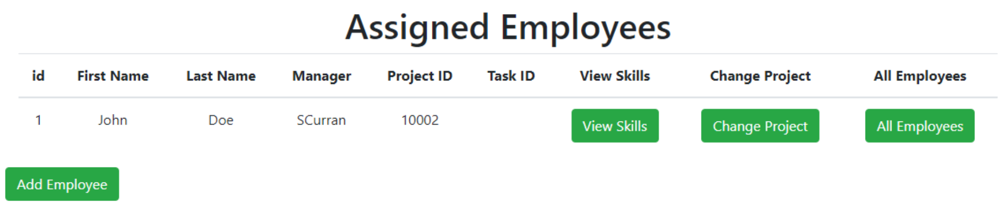


List Projects

id	Description	Manager	Target Date	Completed?	View Employees	View Tasks	Update	Delete
10001	Covid 19 App	SCurran	2021-05-08	false	View Employees	View Tasks	Update	Delete
10002	Bug Tracker App	SCurran	2021-05-08	false	View Employees	View Tasks	Update	Delete

Figure 48

Test - Output:



Assigned Employees

id	First Name	Last Name	Manager	Project ID	Task ID	View Skills	Change Project	All Employees
1	John	Doe	SCurran	10002		View Skills	Change Project	All Employees

Add Employee

Figure 49

Test - Result: Pass

Only relevant assigned employees are showing.

Test Case 10 – Testing security.

Objective: To make sure app security prevents user from simply putting in link and accessing page without logging in.

Inputs: localhost:4200/projects

Expected Output: App will simply refresh login page and prevent any access without logging in.

Test – Input:

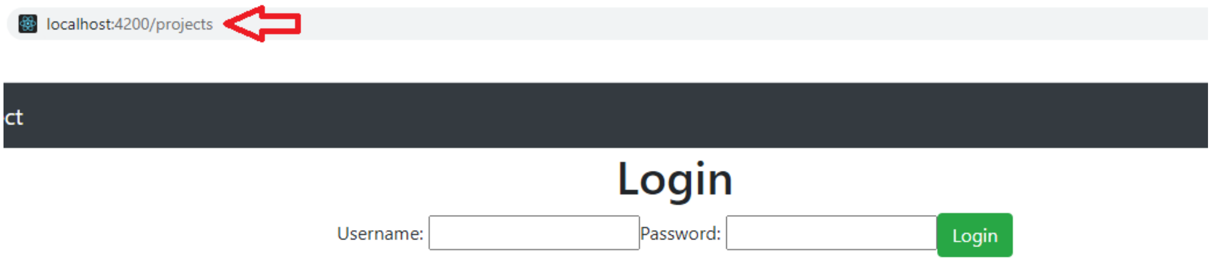


Figure 50

Test - Output:

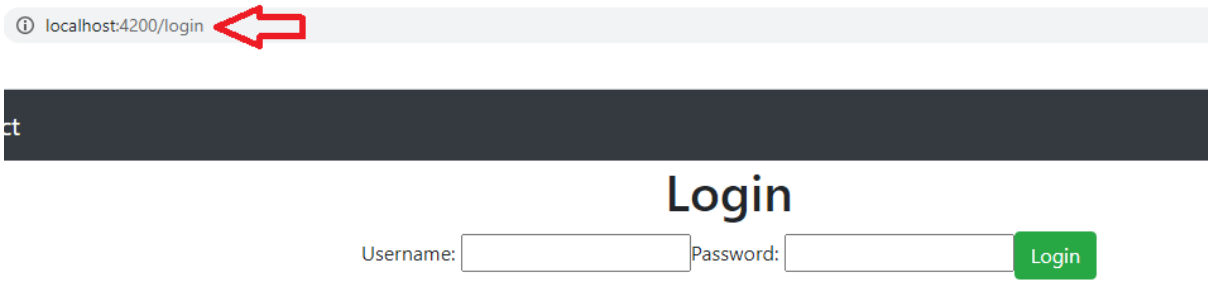


Figure 51

Test - Result: **Pass**

The page cannot be accessed without logging in as expected.

Test Case 11 – Try to delete without admin privileges.

Objective: To see if standard user account is restricted from deleting as intended.

Inputs: username, password, login (button click), Projects tab (click), Delete (button click

Expected Output: User is prevented from deleting project.

Test – Input 1:

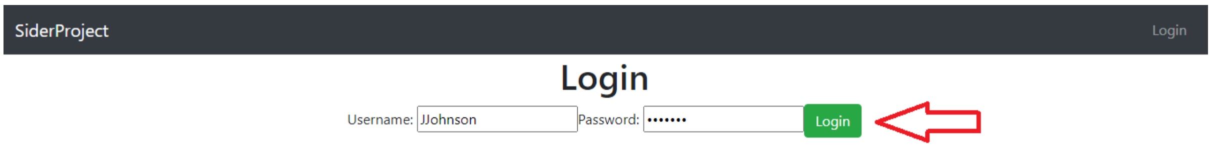
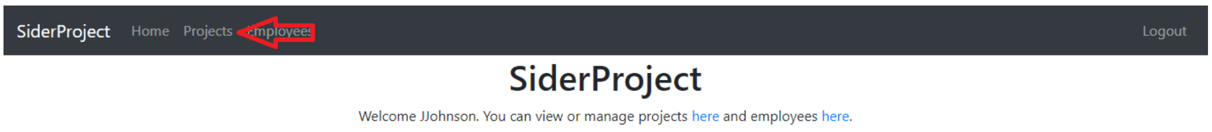


Figure 52

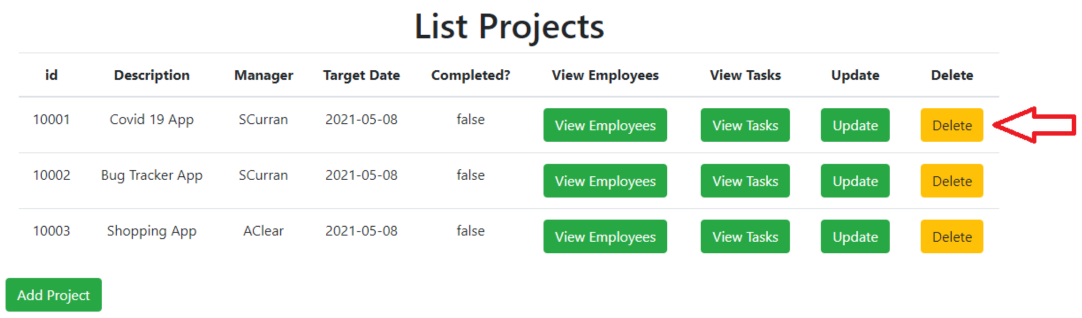
Test – Input 2:



Welcome JJohnson. You can view or manage projects here and employees here.

Figure 53

Test – Input 3:

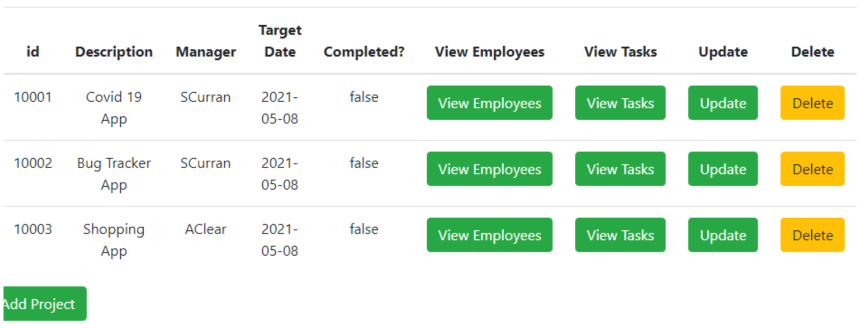


List Projects

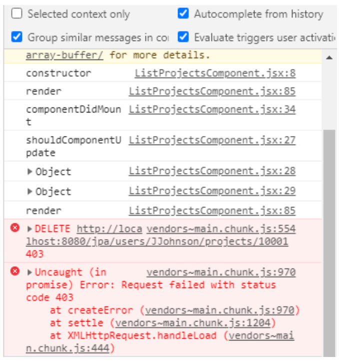
id	Description	Manager	Target Date	Completed?	View Employees	View Tasks	Update	Delete
10001	Covid 19 App	SCurran	2021-05-08	false	View Employees	View Tasks	Update	Delete
10002	Bug Tracker App	SCurran	2021-05-08	false	View Employees	View Tasks	Update	Delete
10003	Shopping App	AClear	2021-05-08	false	View Employees	View Tasks	Update	Delete

Figure 54

Test - Output:



id	Description	Manager	Target Date	Completed?	View Employees	View Tasks	Update	Delete
10001	Covid 19 App	SCurran	2021-05-08	false	View Employees	View Tasks	Update	Delete
10002	Bug Tracker App	SCurran	2021-05-08	false	View Employees	View Tasks	Update	Delete
10003	Shopping App	AClear	2021-05-08	false	View Employees	View Tasks	Update	Delete



☐ Selected context only ☑ Autocomplete from history
☑ Group similar messages in co ☑ Evaluate triggers user activati

```
array-buffer/ for more details.
constructor        ListProjectsComponent.jsx:8
render             ListProjectsComponent.jsx:85
componentDidMoun   ListProjectsComponent.jsx:34
t
shouldComponentU   ListProjectsComponent.jsx:27
pdate
▸Object            ListProjectsComponent.jsx:28
▸Object            ListProjectsComponent.jsx:29
render             ListProjectsComponent.jsx:85
▸DELETE http://loca vendors~main.chunk.js:554
lhost:8080/jpa/users/JJohnson/projects/10001
403
▸Uncaught (in       vendors~main.chunk.js:970
promise) Error: Request failed with status
code 403
    at createError (vendors~main.chunk.js:970)
    at settle (vendors~main.chunk.js:1204)
    at XMLHttpRequest.handleLoad (vendors~mai
n.chunk.js:444)
```

Test - Result: Pass

An error in console displays prevented user from deleting due to lack of admin privileges.

Final Verdict

All main requirements work as intended and the tests have proven successful.

Test Case 1	***Testing Login with valid details.***	Pass
Test Case 2	***Testing Login with invalid details.***	Pass
Test Case 3	***Fetching all projects from database.***	Pass
Test Case 4	***Add new project to list.***	Pass
Test Case 5	***Update a current project.***	Pass
Test Case 6	***Delete a current project.***	Pass
Test Case 7	***Add task to a specific project.***	Pass
Test Case 8	***Add Employee to employees list and assign them to specific project.***	Pass
Test Case 9	***Check that employees for specified project are displaying.***	Pass
Test Case 10	***Testing Security.***	Pass
Test Case 11	***Try to delete without admin privileges.***	Pass

7. Conclusion

Overall the project had fulfilled the students main requirements in some capacity. There were plenty of bumps in the road along the way but the student in time eventually overcame them. The project plan (see appendix 4) was mostly followed but not fully due to occasional delays setting the student back. Due to some help and encouragement from his advisor, the student was always able to get back on track. Overall the project proved to be an educational and enjoyable experience for the student.

7.1 What did the student learn?

One area the student received a thorough understanding of was REST in particular mapping and the GET, PUT, POST and DELETE features. This process of course had to be repeated several times for the different Project, Employee, Task and Skill objects. REST would eventually feature in the student's Enterprise Java module which meant the student went into to both the portion of the module and the eventual assignment with an early advantage. The student also learned of the many features of Java Spring in particular hibernate. The student also now has a basic understand and experience with the ReactJS framework as well as the Visual Studio Code IDE, in particular the use of components and the overall convenience of many aspects of the framework over standard javascript. All of these learning experiences will be useful when the student starts their internship as both Java and REACT are languages and frameworks used there. Overall the student has put their foot in the door of the full stack world and while there is still much more to learn, they have taken the major first step to become a full stack developer.

7.2 What could be improved?

How logins work could do with being revamped as currently all logins are hardcoded. There was initially plans to create a register button to the app where an account could be created but due to time constraints, this was never implemented. This would have included an option for an admin to assign another login admin status. Also when assigning employees to projects, an option where you could have added them by checkbox could have looked much more impressive.

8. Bibliography

1. Java Brains (2011) *Maven Tutorial Playlist* – Available at: https://www.youtube.com/watch?v=al7bRZzz4oU&list=PL92E8 (Accessed 8th February 2021)
2. freeCodeCamp (2020) *The React Beginners Handbook* – Available at: https://www.freecodecamp.org/news/react-beginner-handbook/ (Accessed 9th February 2021)
3. Codeevolution (2018) *ReactJS Tutorial* – Available at: https://www.youtube.com/watch?v=QFaFIcGhPoM&list=PLC3y8 (Accessed 9th February 2021)
4. Mukund Madhav – Medium (2020) *full-stack app* – Available at: https://mukundmadhav.medium.com/build-and-deploy-react-app-with-spring-boot-and-mysql-6f888eb0c600 (Accessed 15th February 2021)
5. Simplilearn (2021) *Most Widely Used Project Management Apps Compared* – Available at: https://www.simplilearn.com/project-management-apps-article (Accessed 17th February 2021)
6. Neil Patel (2021) *Best Employee Scheduling Software* – Available at: https://neilpatel.com/blog/best-employee-scheduling-software/ (Accessed 17th February 2021)
7. Ranga Karanam (2019) *Todo – Springboot Tutorial* – Available at: https://www.springboottutorial.com/ (Accessed 1st March 2021)
8. Ranga Karanam (2020) *Unit Testing Rest Services with Spring Boot and Junit* – Available at: https://www.springboottutorial.com/unit-testing-for-spring-boot-rest-services (Accessed 7th April 2021)

9. Appendices

Appendix 1: Learning Journals

Week 1

1. What tasks have you undertaken in your project this week?

My first task was getting all of the tools ready. I checked to see did my Eclipse IDE have Maven builds (which it did) and made a test project. I downloaded the Visual Studio Code IDE which I will be using for react. I created a test react project as well. I also watched a few youtube videos on both Maven and React, particularly from the Java Brains channel recommended to me by my internship. So in terms of tools and technologies, i'm ready to go. I am also close to set on the project idea which is currently a Project Management and Employee Scheduling app which I discussed with my supervisor.

2. What challenges have you encountered or in any way prevented you from progressing those tasks?

In terms of requirements, specifications and the project idea itself, I am not 100% solidified on this yet. As per my supervisor, I emailed my internship to see can we discuss this and I am awaiting their response. Basically until the requirements, specifications and idea are fully decided on by both myself and my internship, my project cannot progress properly.

3. What are you going to do next week to resolve those tasks?

Hopefully speak to my internship to finally be fully set on the idea, requirements and specifications. If I cannot get speaking to them, I will go with my current idea and try to start building a foundation for an end to end full stack CRUD app which is flexible enough to make changes should my project idea, requirements or specifications change. My internship already confirmed they would like to see an end to end app using java (with maven builds) with a database and frontend of my choice, but preferably react which they use. So there should be no issues starting off with a template even if the idea hasn't been fully established yet

Week 2

1. What tasks have you undertaken in your project this week?

Officially started the base of the CRUD app. Spoke to Sidero in a meeting Friday morning who have established they are happy with me to go my own way on requirements and specifications. In terms of ideas, they are happy with any management system which I suggested. They showed an example of a skills management system I can do but they are perfectly happy with the project management and employee scheduling idea as long as I become competent in the technologies and tools they use.

2. What challenges have you encountered or in any way prevented you from progressing those tasks?

Until the meeting with Sidero, I was still not 100% on the idea, requirements or specifications. The newly suggested idea of a skills management app has also made me rethink the idea overall. I will stick with my original idea although I could possibly implement a skills section on the app.

3. What are you going to do next week to resolve those tasks?

Continue with and finish my requirements document now that I established where Sidero are at. Also continue building the base of the app.

Week 3

1. What tasks have you undertaken in your project this week?

The main task this week was the requirements document. I also got started on the project design later in the week.

2. What challenges have you encountered or in any way prevented you from progressing those tasks?

Power outage happened so I missed my initial meeting with my NUIG supervisor on Tuesday and it was delayed until Thursday. Got feedback in areas the requirements document could be improved. I then used that feedback to fix the document.

3. What are you going to do next week to resolve those tasks?

Continue with the project design now that the requirements document will be out of the way. I will now try to adhere to my project plan and also get started officially on the projects frontend.

Week 4

1. What tasks have you undertaken in your project this week?

The project design and the front end development. I've started with the login form.

2. What challenges have you encountered or in any way prevented you from progressing those tasks?

Trying to juggle between the main project and two other assignments. I needed to finish my Enterprise Java assignment at the start of the week and get started on the Object Oriented assignment.

3. What are you going to do next week to resolve those tasks?

Try to get the other assignments out of the way in the weekend which leaves me a whole week to get the front end worked on and hopefully finished.

Week 5

1. What tasks have you undertaken in your project this week?

Working primarily on the front end for the project. In particular the security login screen which then forwards to the home page.

2. What challenges have you encountered or in any way prevented you from progressing those tasks?

There were several issues installing dependencies for my projects package.json in visual studio code. This was eventually rectified however. I am also finding myself getting a bit behind schedule.

3. What are you going to do next week to resolve those tasks?

Now that my other assignments are finished, i'll use the weekend for catch up. The next time I receive assignments, the intention is to tackle them quickly freeing up time for the main project.

Week 6

1. What tasks have you undertaken in your project this week?

Working on java springboot with maven and connecting react to RESTful api.

2. What challenges have you encountered or in any way prevented you from progressing those tasks?

Some issues with Maven in eclipse and an assignment in another module held things up.

3. What are you going to do next week to resolve those tasks?

Issues with Maven were resolved and assignment was completed freeing up plenty of free time for next week to work on RESTful api.

Week 7

1. What tasks have you undertaken in your project this week?

Continued working on connecting the front end to backend and working on Spring security.

2. What challenges have you encountered or in any way prevented you from progressing those tasks?

An issue with CORS policy when working on Spring security.

3. What are you going to do next week to resolve those tasks?

Testing the API with API tester with ever change to avoid CORS policy.

Week 8

1. What tasks have you undertaken in your project this week?

Finished connecting front end to back end. Worked on security.

2. What challenges have you encountered or in any way prevented you from progressing those tasks?

More issues with CORS when working on security.

3. What are you going to do next week to resolve those tasks?

Using API tester on every step. Watching youtuve videos on Java Spring security and how to prevent these issues in the future.

Week 9

1. What tasks have you undertaken in your project this week?

Finished the body of the project. The project now has full CRUD functionality and a H2 database. Now I need to work on getting most of the deliverables.

2. What challenges have you encountered or in any way prevented you from progressing those tasks?

Some issues with the database initially but this was fixed. Other personal issues also hindered me this week, in particular the sending of documentation to my Academic supervisor.

3. What are you going to do next week to resolve those tasks?

Get started on some documentation and send it off before my next meeting on Tuesday.

Week 10

1. What tasks have you undertaken in your project this week?

Attempting to expand the project further by adding tasks to each indivudal project and connecting the project list to the employee list.

2. What challenges have you encountered or in any way prevented you from progressing those tasks?

I'm running into issues on adding tasks part. I'm getting a [object%20Object] in the path and i'm trying to understand why. Also lots on in promise errors.

3. What are you going to do next week to resolve those tasks?

All I can do is keeping trying. I have to thoroughly check over both the front end and back end files for the tasks.

Week 11

1. What tasks have you undertaken in your project this week?

Connecting the employees list to the projects list and getting started on some testing.

2. What challenges have you encountered or in any way prevented you from progressing those tasks?

Any employees with a null project id will not show in main project page. Project ID itself also wasn't displaying on frontend until I fixed it. This has delayed testing as I was waiting until I finished connecting project to employee before proceeding.

3. What are you going to do next week to resolve those tasks?

Try to finish all main tasks by the end of this week. The admin and employee logins still need to be started.

Week 12

1. What tasks have you undertaken in your project this week?

Finished connecting employee list to project list and also connect employees to tasks. Started on different viewings for an admin and employee login.

2. What challenges have you encountered or in any way prevented you from progressing those tasks?

Errors in trying to delete entities from certain lists without removing entities from other lists. Eventually fixed this. Some errors in security too for the different logins.

3. What are you going to do next week to resolve those tasks?

Finish the different logins which means all main objectives complete and tidy up app where possible. I also need to really start working on the document as minimal of it is done.

Week 13

1. What tasks have you undertaken in your project this week?

Finished all main objectives in terms of coding. The final objective is to finish my document accompanying the project.

2. What challenges have you encountered or in any way prevented you from progressing those tasks?

After various attempts, I was unable to figure out a way to hide the frontend add/update/delete buttons specifically from non-admins so I will have to make do with the buttons showing with the user simply being prevented from using them by getting an error.

3. What are you going to do next week to resolve those tasks?

There are no more tasks beyond finishing the document. By next week, the visio will have happened showing off my finished product making this the final journal entry.

Appendix 2: Project Definition Document

https://github.com/StevenCurranBlacklion/SiderProject/blob/main/Documents/Project%20Definition%20Document.pdf

Appendix 3: Use Case Descriptions

https://github.com/StevenCurranBlacklion/SiderProject/blob/main/Documents/Use%20Case%20Descriptions.pdf

Appendix 4: Project Plan (from Project Definition Document)

Week	Start Date	Actions
Week 1	08/02/2021	• Determine technologies with NUIG advisor and Sidero advisor. • Watch videos and read about unused technologies.
Week 2	15/02/2021	• Determine idea with NUIG advisor and Sidero advisor. • Practice with technologies.
Week 3	22/02/2021	• Design Requirements document. • Get started on System Design.
Week 4	01/03/2021	• Finish System Design. • Get started on frontend using react.
Week 5	08/03/2021	• Combine react and Spring Boot to begin full stack application
Week 6	15/03/2021	• Beginning of Spring Boot REST API.
Week 7	22/03/2021	• Connect react UI to RESTful API. • Show off prototype to advisors.
Week 8	29/03/2021	• Add security to application. • Create login / register UI.
Week 9	05/04/2021	• Implement JWT framework. • Make database.
Week 10	12/04/2021	• Integrate backend using JPA and Hibernate. • Finish essential deliverables.
Week 11	19/04/2021	• Work on desirable deliverables. • Begin Test and Design document.
Week 12	26/04/2021	• Work on desirable deliverables. • Finish Test and Design document. • Have project ready for presentation.