MySQL download-install-setup:

<https://dev.mysql.com/downloads/file/?id=497106>

<https://www.mysqltutorial.org/install-mysql/#:~:text=Install%20MySQL%20via%20MySQL%20Installer&text=Install%20MySQL%20Step%203%20%E2%80%93%20Download,server%2C%20MySQL%20Workbench%2C%20etc.&text=Install%20MySQL%20Step%205%20%E2%80%93%20Choosing,are%20several%20setup%20types%20available.>

<https://www.youtube.com/watch?v=X_umYKqKaF0>

Project:

1. Create a SpringBoot project named studentboot with web, Spring Data JPA, SpringBoot Dev Tools and MySQL Server Driver packages. Extract that project.
2. Import the project in Eclipse.
3. Create the configuration class  
   Instead of XML, we perform annotation-based configuration. So, we create a class Config.java and specify the required configuration in it. However, there is one more configuration class StudentApplication.java. This class is provided by Spring Boot automatically.

**package** com.niit.studentboot.config;

**import** java.util.Properties;

**import** javax.sql.DataSource;

**import** org.springframework.beans.factory.annotation.Value;

**import** org.springframework.boot.autoconfigure.EnableAutoConfiguration;

**import** org.springframework.boot.autoconfigure.orm.jpa.HibernateJpaAutoConfiguration;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.ComponentScan;

**import** org.springframework.context.annotation.ComponentScans;

**import** org.springframework.context.annotation.Configuration;

**import** org.springframework.jdbc.datasource.DriverManagerDataSource;

**import** org.springframework.orm.hibernate5.HibernateTransactionManager;

**import** org.springframework.orm.hibernate5.LocalSessionFactoryBean;

**import** org.springframework.transaction.annotation.EnableTransactionManagement;

**import** org.springframework.web.servlet.view.InternalResourceViewResolver;

@Configuration

@EnableTransactionManagement

@EnableAutoConfiguration(exclude = { HibernateJpaAutoConfiguration.**class**})

@ComponentScans(value = { @ComponentScan("com.niit.studentboot"),

@ComponentScan("Model"),

@ComponentScan("Controller"),

@ComponentScan("DAO"),

@ComponentScan("Miscallaneous"),

@ComponentScan("Service")})

**public** **class** Config {

@Value("${db.driver}")

**private** String DB\_DRIVER;

@Value("${db.password}")

**private** String DB\_PASSWORD;

@Value("${db.url}")

**private** String DB\_URL;

@Value("${db.username}")

**private** String DB\_USERNAME;

@Value("${hibernate.dialect}")

**private** String HIBERNATE\_DIALECT;

@Value("${hibernate.show\_sql}")

**private** String HIBERNATE\_SHOW\_SQL;

@Value("${hibernate.hbm2ddl.auto}")

**private** String HIBERNATE\_HBM2DDL\_AUTO;

@Value("${entitymanager.packagesToScan}")

**private** String ENTITYMANAGER\_PACKAGES\_TO\_SCAN;

@Bean

**public** LocalSessionFactoryBean sessionFactory() {

LocalSessionFactoryBean sessionFactory = **new** LocalSessionFactoryBean();

sessionFactory.setDataSource(dataSource());

sessionFactory.setPackagesToScan(ENTITYMANAGER\_PACKAGES\_TO\_SCAN);

Properties hibernateProperties = **new** Properties();

hibernateProperties.put("hibernate.dialect", HIBERNATE\_DIALECT);

hibernateProperties.put("hibernate.show\_sql", HIBERNATE\_SHOW\_SQL);

hibernateProperties.put("hibernate.hbm2ddl.auto", HIBERNATE\_HBM2DDL\_AUTO);

sessionFactory.setHibernateProperties(hibernateProperties);

**return** sessionFactory;

}

@Bean

**public** DataSource dataSource() {

DriverManagerDataSource dataSource = **new** DriverManagerDataSource();

dataSource.setDriverClassName(DB\_DRIVER);

dataSource.setUrl(DB\_URL);

dataSource.setUsername(DB\_USERNAME);

dataSource.setPassword(DB\_PASSWORD);

**return** dataSource;

}

@Bean

**public** HibernateTransactionManager transactionManager() {

HibernateTransactionManager txManager = **new** HibernateTransactionManager();

txManager.setSessionFactory(sessionFactory().getObject());

**return** txManager;

}

@Bean

**public** InternalResourceViewResolver jspViewResolver() {

InternalResourceViewResolver resolver= **new** InternalResourceViewResolver();

resolver.setPrefix("/views/");

resolver.setSuffix(".jsp");

**return** resolver;

}

}

1. Create the entity class  
   Here, we are creating an Entity/POJO (Plain Old Java Object) class.

**package** com.niit.studentboot.model;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

@Entity

@Table(name="Student")

**public** **class** Student {

@Id

@GeneratedValue(strategy=GenerationType.***IDENTITY***)

**private** **int** student\_id;

**private** String student\_name;

**private** String student\_email;

**private** String student\_branch;

**public** **int** getStudent\_id() {

**return** student\_id;

}

**public** **void** setStudent\_id(**int** student\_id) {

**this**.student\_id = student\_id;

}

**public** String getStudent\_name() {

**return** student\_name;

}

**public** **void** setStudent\_name(String student\_name) {

**this**.student\_name = student\_name;

}

**public** String getStudent\_email() {

**return** student\_email;

}

**public** **void** setStudent\_email(String student\_email) {

**this**.student\_email = student\_email;

}

**public** String getStudent\_branch() {

**return** student\_branch;

}

**public** **void** setStudent\_branch(String student\_branch) {

**this**.student\_branch = student\_branch;

}

}

1. Create the DAO interface implementation class

**package** com.niit.studentboot.dao;

**import** java.util.List;

**import** com.niit.studentboot.model.Student;

**public** **interface** Student\_DAO {

**public** **boolean** saveStudent(Student student);

**public** List<Student> getStudents();

**public** **boolean** deleteStudent(Student student);

**public** List<Student> getStudentByID(Student student);

**public** **boolean** updateStudent(Student student);

}

1. Create the DAO interface implementation class

**package** com.niit.studentboot.dao;

**import** java.util.List;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.query.Query;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Repository;

**import** com.niit.studentboot.model.Student;

@Repository

**public** **class** Student\_DAO\_Impl **implements** Student\_DAO{

@Autowired

**private** SessionFactory sessionFactory;

@Override

**public** **boolean** saveStudent(Student student) {

**boolean** status=**false**;

**try** {

sessionFactory.getCurrentSession().save(student);

status=**true**;

} **catch** (Exception e) {

e.printStackTrace();

}

**return** status;

}

@Override

**public** List<Student> getStudents() {

Session currentSession = sessionFactory.getCurrentSession();

Query<Student> query=currentSession.createQuery("from Student", Student.**class**);

List<Student> list=query.getResultList();

**return** list;

}

@Override

**public** **boolean** deleteStudent(Student student) {

**boolean** status=**false**;

**try** {

sessionFactory.getCurrentSession().delete(student);

status=**true**;

} **catch** (Exception e) {

e.printStackTrace();

}

**return** status;

}

@Override

**public** List<Student> getStudentByID(Student student) {

Session currentSession = sessionFactory.getCurrentSession();

Query<Student> query=currentSession.createQuery("from Student where student\_id=:student\_id", Student.**class**);

query.setParameter("student\_id", student.getStudent\_id());

List<Student> list=query.getResultList();

**return** list;

}

@Override

**public** **boolean** updateStudent(Student student) {

**boolean** status=**false**;

**try** {

sessionFactory.getCurrentSession().update(student);

status=**true**;

} **catch** (Exception e) {

e.printStackTrace();

}

**return** status;

}

}

1. Create the service layer interface

Here, we are creating a service layer interface that acts as a bridge between DAO and Entity classes.

**package** com.niit.studentboot.service;

**import** java.util.List;

**import** com.niit.studentboot.model.Student;

**public** **interface** Student\_Service {

**public** **boolean** saveStudent(Student student);

**public** List<Student> getStudents();

**public** **boolean** deleteStudent(Student student);

**public** List<Student> getStudentByID(Student student);

**public** **boolean** updateStudent(Student student);

}

1. Create the service layer implementation class

**package** com.niit.studentboot.service;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.niit.studentboot.dao.Student\_DAO;

**import** com.niit.studentboot.model.Student;

@Service

@Transactional

**public** **class** Student\_Service\_Impl **implements** Student\_Service {

@Autowired

**private** Student\_DAO studentdao;

@Override

**public** **boolean** saveStudent(Student student) {

**return** studentdao.saveStudent(student);

}

@Override

**public** List<Student> getStudents() {

**return** studentdao.getStudents();

}

@Override

**public** **boolean** deleteStudent(Student student) {

**return** studentdao.deleteStudent(student);

}

@Override

**public** List<Student> getStudentByID(Student student) {

**return** studentdao.getStudentByID(student);

}

@Override

**public** **boolean** updateStudent(Student student) {

**return** studentdao.updateStudent(student);

}

}

1. Create the controller class

**package** com.niit.studentboot.controller;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.web.bind.annotation.CrossOrigin;

**import** org.springframework.web.bind.annotation.DeleteMapping;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.PostMapping;

**import** org.springframework.web.bind.annotation.RequestBody;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

**import** com.niit.studentboot.model.Student;

**import** com.niit.studentboot.service.Student\_Service;

@RestController

@CrossOrigin(origins="http://localhost:4200")

@RequestMapping(value="/api")

**public** **class** StudentController {

@Autowired

**private** Student\_Service studentservice;

@PostMapping("save-student")

**public** **boolean** saveStudent(@RequestBody Student student) {

**return** studentservice.saveStudent(student);

}

@GetMapping("students-list")

**public** List<Student> allstudents() {

**return** studentservice.getStudents();

}

@DeleteMapping("delete-student/{student\_id}")

**public** **boolean** deleteStudent(@PathVariable("student\_id") **int** student\_id,Student student) {

student.setStudent\_id(student\_id);

**return** studentservice.deleteStudent(student);

}

@GetMapping("student/{student\_id}")

**public** List<Student> allstudentByID(@PathVariable("student\_id") **int** student\_id,Student student) {

student.setStudent\_id(student\_id);

**return** studentservice.getStudentByID(student);

}

@PostMapping("update-student/{student\_id}")

**public** **boolean** updateStudent(@RequestBody Student student,@PathVariable("student\_id") **int** student\_id) {

student.setStudent\_id(student\_id);

**return** studentservice.updateStudent(student);

}

}

1. Edit application.properties file  
   Here, we are editing the **application.properties** file present inside the **src/main/resources** folder. The following file contains the configuration properties.

# Database

db.driver=com.mysql.jdbc.Driver

db.url= jdbc:mysql://localhost:3306/indigo

db.username=root

db.password=niit@123

# Hibernate

hibernate.dialect=org.hibernate.dialect.MySQL5Dialect

hibernate.show\_sql=true

hibernate.hbm2ddl.auto=update

entitymanager.packagesToScan=com.niit.studentboot.model

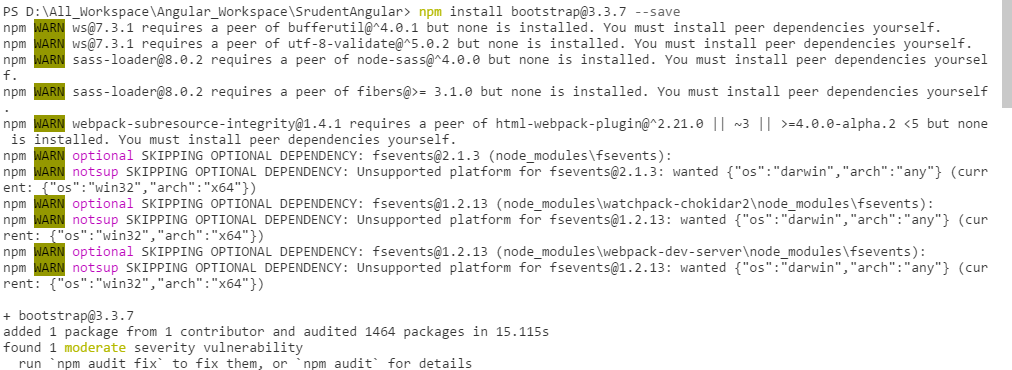
1. Open Visual Studio Code. Create new project inside Angular\_Workspace. Set the project name as StudentAngular

ng new StudentAngular

1. Install Bootstrap CSS framework

Use the following command to install bootstrap in the project.

*D:\All\_Workspace\Angular\_Workspace\SrudentAngular>* ***npm install*** [***bootstrap@3.3.7***](mailto:bootstrap@3.3.7) ***--save***



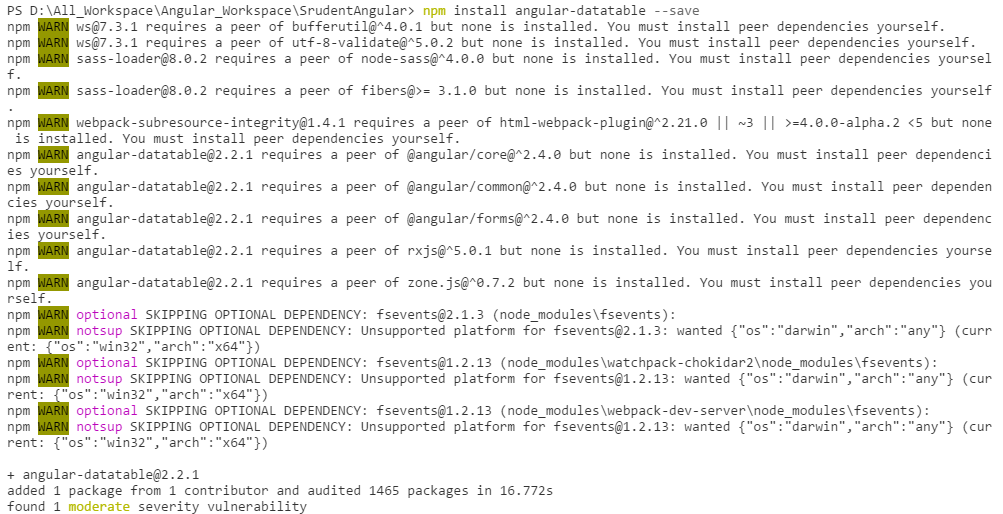
1. Now, include the following code in the style.css file.

@import "~bootstrap/dist/css/bootstrap.css";

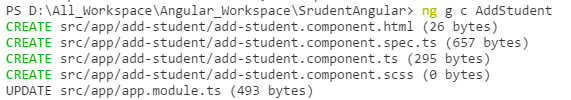
1. Install Angular Data Table

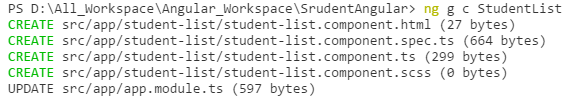
Use the following command to install bootstrap in the project.

*D:\All\_Workspace\Angular\_Workspace\SrudentAngular>* ***npm install angular-datatable --save***



1. It is required to include **DataTableModule** in imports array of **app.module.ts** file.
2. Generate Components  
   Open the project in visual studio and then use the following command to generate Angular components:  
   *ng g c AddStudent*  
   *ng g c StudentList*





1. Let's also create a service class by using the following command: -

ng g s Student



1. Edit the **app.module.ts** file
   1. **Import Routing** - Here, we are importing routing file (app-routing.module.ts) and include it in imports array.
   2. **Import ReactiveFormsModule** - Here, we are importing **ReactiveFormsModule** for reactive forms and specify it in imports array.
   3. **Import HttpModule** - Here, we are importing **HttpModule** for server requests and specifying it in imports array.
   4. **Register Service class** - Here, we are mentioning the service class in provider's array.

//app.module.ts

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import { AppRoutingModule } from './app-routing.module';

import { AppComponent } from './app.component';

import { FormsModule, ReactiveFormsModule } from '@angular/forms';

import { HttpClientModule } from '@angular/common/http';

import {DataTablesModule} from 'angular-datatables';

import { AddStudentComponent } from './add-student/add-student.component';

import { StudentListComponent } from './student-list/student-list.component';

@NgModule({

declarations: [

AppComponent,

AddStudentComponent,

StudentListComponent

],

imports: [

BrowserModule,

AppRoutingModule,

FormsModule,

ReactiveFormsModule,

HttpClientModule,

DataTablesModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

1. Edit the **app-routing.module.ts** file

import { NgModule } from '@angular/core';

import { Routes, RouterModule } from '@angular/router';

import { StudentListComponent } from './student-list/student-list.component';

import { AddStudentComponent } from './add-student/add-student.component';

const routes: Routes = [

{ path: '', redirectTo: 'view-student', pathMatch: 'full' },

{ path: 'view-student', component: StudentListComponent },

{ path: 'add-student', component: AddStudentComponent },

];

@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule]

})

export class AppRoutingModule { }

1. Edit the **app.component.html** file(remove existing code and replace with following code)

<div class="container-fluid">

<nav class="navbar navbar-expand-sm bg-dark navbar-dark">

<ul class="navbar-nav">

<li class="nav-item ">

<a routerLink="view-student" class="nav-link" class="btn btn-primary active" role="button" >View Student</a>

</li>

<li class="nav-item">

<a routerLink="add-student" class="nav-link" class="btn btn-primary active" role="button" >Add Student</a>

</li>

</ul>

</nav>

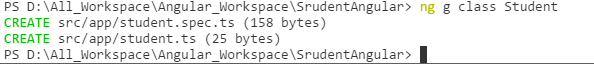
<router-outlet></router-outlet>

</div>

1. Create the **Student.ts** class

Let's create a class by using the following command: -

*ng g class Student*



1. Now, specify the required fields within the **Student** class. (The purpose of this class is to map the specified fields with the fields of Spring entity class.)

export class Student {

student\_id:number;

student\_name:String;

student\_email:String;

student\_branch:String;

}

1. Edit the **student.service.ts** file

import { Injectable } from '@angular/core';

import { HttpClient } from '@angular/common/http';

import { Observable } from 'rxjs';

@Injectable({

providedIn: 'root'

})

export class StudentService {

private baseUrl = 'http://localhost:8080/api/';

constructor(private http:HttpClient) { }

getStudentList(): Observable<any> {

return this.http.get(`${this.baseUrl}`+'students-list');

}

createStudent(student: object): Observable<object> {

return this.http.post(`${this.baseUrl}`+'save-student', student);

}

deleteStudent(id: number): Observable<any> {

return this.http.delete(`${this.baseUrl}/delete-student/${id}`, { responseType: 'text' });

}

getStudent(id: number): Observable<Object> {

return this.http.get(`${this.baseUrl}/student/${id}`);

}

updateStudent(id: number, value: any): Observable<Object> {

return this.http.post(`${this.baseUrl}/update-student/${id}`, value);

}

}

1. Edit the **add-student.component.ts** file

import { Component, OnInit } from '@angular/core';

import { StudentService } from '../student.service';

import {FormControl,FormGroup,Validators} from '@angular/forms';

import { Student } from '../student';

@Component({

selector: 'app-add-student',

templateUrl: './add-student.component.html',

styleUrls: ['./add-student.component.scss']

})

export class AddStudentComponent implements OnInit {

constructor(private studentservice:StudentService) { }

student : Student=new Student();

submitted = false;

ngOnInit() {

this.submitted=false;

}

studentsaveform=new FormGroup({

student\_name:new FormControl('' , [Validators.required , Validators.minLength(5) ] ),

student\_email:new FormControl('',[Validators.required,Validators.email]),

student\_branch:new FormControl()

});

saveStudent(saveStudent){

this.student=new Student();

this.student.student\_name=this.StudentName.value;

this.student.student\_email=this.StudentEmail.value;

this.student.student\_branch=this.StudentBranch.value;

this.submitted = true;

this.save();

}

save() {

this.studentservice.createStudent(this.student)

.subscribe(data => console.log(data), error => console.log(error));

this.student = new Student();

}

get StudentName(){

return this.studentsaveform.get('student\_name');

}

get StudentEmail(){

return this.studentsaveform.get('student\_email');

}

get StudentBranch(){

return this.studentsaveform.get('student\_branch');

}

addStudentForm(){

this.submitted=false;

this.studentsaveform.reset();

}

}

1. Edit the **add-student.component.html** file

<h3>Create Student</h3>

<div class="row">

<div class="col-sm-4"></div>

<div class="col-sm-4" >

<div [hidden]="submitted" style="width: 400px;">

<form [formGroup]="studentsaveform" #savestudent (ngSubmit)="saveStudent(saveStudent)">

<div class="form-group">

<label for="name">Student Name</label>

<input type="text" class="form-control" formControlName="student\_name" data-toggle="tooltip"

data-placement="right" title="Enter Student Name" >

<div class="alert alert-danger" \*ngIf = "(StudentName.touched) && (StudentName.invalid)"

style="margin-top: 5px;">

<span \*ngIf="StudentName.errors.required">Student Name is Required</span>

<span \*ngIf = "StudentName.errors.minlength">

MinLength Error

</span>

</div>

</div>

<div class="form-group">

<label for="name">Student Email</label>

<input type="text" class="form-control" formControlName="student\_email"

data-toggle="tooltip" data-placement="right" title="Enter Student Email">

<div class="alert alert-danger" \*ngIf = "(StudentEmail.touched) && (StudentEmail.invalid)"

style="margin-top: 5px;">

<span \*ngIf="StudentEmail.errors.required">Student Email is Required</span>

<span \*ngIf = "StudentEmail.errors.email">

Invalid Email Format

</span>

</div>

</div>

<div class="form-group">

<label for="branch">Student Branch</label>

<select class="form-control" formControlName="student\_branch" data-toggle="tooltip"

data-placement="right" title="Select Student Branch">

<option value="null">--Select Branch--</option>

<option value="B-Tech">B-Tech</option>

<option value="BCA">BCA</option>

<option value="MCA">MCA</option>

<option value="M-Tech">M-Tech</option>

</select>

</div>

<button type="submit" class="btn btn-success">Submit</button>

</form>

</div>

</div>

<div class="col-sm-4"></div>

</div>

<div class="row">

<div class="col-sm-4"></div>

<div class="col-sm-4">

<div [hidden]="!submitted">

<h4>Student Added SuccessFully!</h4>

<button (click)="addStudentForm()" class='btn btn-primary'>Add More Student</button>

</div>

</div>

<div class="col-sm-4"></div>

</div>

1. Edit the **student-list.component.ts** file

import { Component, OnInit } from '@angular/core';

import { StudentService } from '../student.service';

import { Student } from '../student';

import { Observable,Subject } from "rxjs";

import { Validators, FormControl, FormGroup, FormBuilder} from '@angular/forms';

@Component({

selector: 'app-student-list',

templateUrl: './student-list.component.html',

styleUrls: ['./student-list.component.scss']

})

export class StudentListComponent implements OnInit {

constructor(private studentservice:StudentService) { }

studentsArray: any[] = [];

dtOptions: DataTables.Settings = {};

dtTrigger: Subject<any>= new Subject();

students: Observable<Student[]>;

student : Student=new Student();

deleteMessage=false;

studentlist:any;

isupdated = false;

ngOnInit() {

this.isupdated=false;

this.dtOptions = {

pageLength: 6,

stateSave:true,

lengthMenu:[[6, 16, 20, -1], [6, 16, 20, "All"]],

processing: true

};

this.studentservice.getStudentList().subscribe(data =>{

this.students =data;

this.dtTrigger.next();

})

}

deleteStudent(id: number) {

this.studentservice.deleteStudent(id)

.subscribe(

data => {

console.log(data);

this.deleteMessage=true;

this.studentservice.getStudentList().subscribe(data =>{

this.students =data

})

},

error => console.log(error));

}

updateStudent(id: number){

this.studentservice.getStudent(id)

.subscribe(

data => {

this.studentlist=data

},

error => console.log(error));

}

studentupdateform=new FormGroup({

student\_id:new FormControl(),

student\_name:new FormControl(),

student\_email:new FormControl(),

student\_branch:new FormControl()

});

updateStu(updstu){

this.student=new Student();

this.student.student\_id=this.StudentId.value;

this.student.student\_name=this.StudentName.value;

this.student.student\_email=this.StudentEmail.value;

this.student.student\_branch=this.StudentBranch.value;

console.log(this.StudentBranch.value);

this.studentservice.updateStudent(this.student.student\_id,this.student).subscribe(

data => {

this.isupdated=true;

this.studentservice.getStudentList().subscribe(data =>{

this.students =data

})

},

error => console.log(error));

}

get StudentName(){

return this.studentupdateform.get('student\_name');

}

get StudentEmail(){

return this.studentupdateform.get('student\_email');

}

get StudentBranch(){

return this.studentupdateform.get('student\_branch');

}

get StudentId(){

return this.studentupdateform.get('student\_id');

}

changeisUpdate(){

this.isupdated=false;

}

}

1. Edit the **student-list.component.html** file

<div class="panel panel-default">

<div class="panel-heading">

<h1 style="text-align: center">Students</h1><br>

<div class="row" [hidden]="!deleteMessage">

<div class="col-sm-4"></div>

<div class="col-sm-4">

<div class="alert alert-info alert-dismissible">

<button type="button" class="close" data-dismiss="alert">×</button>

<strong>Student Data Deleted</strong>

</div>

</div>

<div class="col-sm-4"></div>

</div>

</div>

<div class="panel-body">

<table class="table table-hover table-sm" datatable [dtOptions]="dtOptions"

[dtTrigger]="dtTrigger" >

<thead class="thead-light">

<tr>

<th>Student Name</th>

<th>Student Email</th>

<th>Student Branch</th>

<th>Action</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let student of students ">

<td>{{student.student\_name}}</td>

<td>{{student.student\_email}}</td>

<td>{{student.student\_branch}}</td>

<td><button (click)="deleteStudent(student.student\_id)" class='btn btn-primary'><i class="fa fa-futboll-0">Delete</i></button>

<button (click)="updateStudent(student.student\_id)" class='btn btn-info'

data-toggle="modal" data-target="#myModal">Update</button>

</td>

</tr>

</tbody><br>

</table>

</div>

</div>

<div class="modal" id="myModal">

<div class="modal-dialog">

<div class="modal-content">

<form [formGroup]="studentupdateform" #updstu (ngSubmit)="updateStu(updstu)">

<!-- Modal Header -->

<div class="modal-header">

<h4 class="modal-title" style="text-align: center">Update Student</h4>

</div>

<!-- Modal body -->

<div class="modal-body" \*ngFor="let student of studentlist " >

<div [hidden]="isupdated">

<input type="hidden" class="form-control" formControlName="student\_id" [(ngModel)]="student.student\_id">

<div class="form-group">

<label for="name">Student Name</label>

<input type="text" class="form-control" formControlName="student\_name" [(ngModel)]="student.student\_name" >

</div>

<div class="form-group">

<label for="name">Student Email</label>

<input type="text" class="form-control" formControlName="student\_email" [(ngModel)]="student.student\_email">

</div>

<div class="form-group">

<label for="name">Student Branch</label>

<select class="form-control" formControlName="student\_branch" required>

<option value="B-Tech" [selected]="'B-Tech' == student.student\_branch">B-Tech</option>

<option value="BCA" [selected]="'BCA' == student.student\_branch">BCA</option>

<option value="MCA" [selected]="'MCA' == student.student\_branch" >MCA</option>

<option value="M-Tech" [selected]="'M-Tech' == student.student\_branch">M-Tech</option>

</select>

</div>

</div>

<div [hidden]="!isupdated">

<h4>Student Detail Updated!</h4>

</div>

</div>

<!-- Modal footer -->

<div class="modal-footer" >

<button type="submit" class="btn btn-success" [hidden]="isupdated">Update</button>

<button type="button" class="btn btn-danger" data-dismiss="modal" (click)="changeisUpdate()">Close</button>

</div>

</form>

</div>

</div>

</div>

1. Install following:

npm install jquery --save

npm install datatables.net --save

npm install datatables.net-dt --save

npm install angular-datatables@6.0.0 --save

npm install @types/jquery --save-dev

npm install @types/datatables.net --save-dev

1. Edit angular.json file at line 30. styles and scripts:

"styles": [

"src/styles.scss",

"node\_modules/datatables.net-dt/css/jquery.dataTables.css",

"node\_modules/bootstrap/dist/css/bootstrap.css"

],

"scripts": [

"node\_modules/jquery/dist/jquery.js",

"node\_modules/datatables.net/js/jquery.dataTables.js",

"node\_modules/bootstrap/dist/js/bootstrap.js"

]

1. Save All.
2. Run the SpringBoot Application
3. Run the Angular project

