Source Code:

package com.example.doclogin.controller;  
  
import com.example.doclogin.model.Appointment;  
import com.example.doclogin.repository.AppointmentRepository;  
import java.util.List;  
import java.util.Map;  
import javax.management.AttributeNotFoundException;  
  
@RestController  
@CrossOrigin  
public class AppointmentController {  
 @Autowired  
 private AppointmentRepository appointmentRepository;  
  
 public AppointmentController() {  
 throw new Error("Unresolved compilation problems: \n\tThe declared package \"com.example.doclogin.Controller\" does not match the expected package \"com.example.doclogin.controller\"\n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tRestController cannot be resolved to a type\n\tCrossOrigin cannot be resolved to a type\n\tRequestMapping cannot be resolved to a type\n\tAutowired cannot be resolved to a type\n\tGetMapping cannot be resolved to a type\n\tThe method findAll() is undefined for the type AppointmentRepository\n\tPostMapping cannot be resolved to a type\n\tRequestBody cannot be resolved to a type\n\tThe method save(Appointment) is undefined for the type AppointmentRepository\n\tGetMapping cannot be resolved to a type\n\tResponseEntity cannot be resolved to a type\n\tPathVariable cannot be resolved to a type\n\tThe method findById(Long) is undefined for the type AppointmentRepository\n\tResponseEntity cannot be resolved\n\tDeleteMapping cannot be resolved to a type\n\tResponseEntity cannot be resolved to a type\n\tPathVariable cannot be resolved to a type\n\tThe method findById(Long) is undefined for the type AppointmentRepository\n\tThe method delete(Appointment) is undefined for the type AppointmentRepository\n\tResponseEntity cannot be resolved\n");  
 }  
  
 public List<Appointment> getAllAppointments() {  
 throw new Error("Unresolved compilation problems: \n\tGetMapping cannot be resolved to a type\n\tThe method findAll() is undefined for the type AppointmentRepository\n");  
 }  
  
 public Appointment createAppointment(@RequestBody Appointment var1) {  
 throw new Error("Unresolved compilation problems: \n\tPostMapping cannot be resolved to a type\n\tRequestBody cannot be resolved to a type\n\tThe method save(Appointment) is undefined for the type AppointmentRepository\n");  
 }  
  
 public ResponseEntity<Appointment> getAppointmentById(@PathVariable Long var1) throws AttributeNotFoundException {  
 throw new Error("Unresolved compilation problems: \n\tGetMapping cannot be resolved to a type\n\tResponseEntity cannot be resolved to a type\n\tPathVariable cannot be resolved to a type\n\tThe method findById(Long) is undefined for the type AppointmentRepository\n\tResponseEntity cannot be resolved\n");  
 }  
  
 public ResponseEntity<Map<String, Boolean>> deleteAppointment(@PathVariable Long var1) throws AttributeNotFoundException {  
 throw new Error("Unresolved compilation problems: \n\tDeleteMapping cannot be resolved to a type\n\tResponseEntity cannot be resolved to a type\n\tPathVariable cannot be resolved to a type\n\tThe method findById(Long) is undefined for the type AppointmentRepository\n\tThe method delete(Appointment) is undefined for the type AppointmentRepository\n\tResponseEntity cannot be resolved\n");  
 }  
}

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.example.doclogin.controller;  
  
import com.example.doclogin.model.Medicine;  
import com.example.doclogin.repository.MedicineRepository;  
import java.util.List;  
import java.util.Map;  
import javax.management.AttributeNotFoundException;  
  
@RestController  
@CrossOrigin  
public class MedicineController {  
 @Autowired  
 private MedicineRepository medicineRepository;  
  
 public MedicineController() {  
 throw new Error("Unresolved compilation problems: \n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tRestController cannot be resolved to a type\n\tCrossOrigin cannot be resolved to a type\n\tRequestMapping cannot be resolved to a type\n\tAutowired cannot be resolved to a type\n\tGetMapping cannot be resolved to a type\n\tThe method findAll() is undefined for the type MedicineRepository\n\tPostMapping cannot be resolved to a type\n\tRequestBody cannot be resolved to a type\n\tThe method save(Medicine) is undefined for the type MedicineRepository\n\tGetMapping cannot be resolved to a type\n\tResponseEntity cannot be resolved to a type\n\tPathVariable cannot be resolved to a type\n\tThe method findById(Long) is undefined for the type MedicineRepository\n\tResponseEntity cannot be resolved\n\tPutMapping cannot be resolved to a type\n\tResponseEntity cannot be resolved to a type\n\tPathVariable cannot be resolved to a type\n\tRequestBody cannot be resolved to a type\n\tThe method findById(Long) is undefined for the type MedicineRepository\n\tThe method save(Medicine) is undefined for the type MedicineRepository\n\tResponseEntity cannot be resolved\n\tDeleteMapping cannot be resolved to a type\n\tResponseEntity cannot be resolved to a type\n\tPathVariable cannot be resolved to a type\n\tThe method findById(Long) is undefined for the type MedicineRepository\n\tThe method delete(Medicine) is undefined for the type MedicineRepository\n\tResponseEntity cannot be resolved\n");  
 }  
  
 public List<Medicine> getAllMedicines() {  
 throw new Error("Unresolved compilation problems: \n\tGetMapping cannot be resolved to a type\n\tThe method findAll() is undefined for the type MedicineRepository\n");  
 }  
  
 public Medicine createMedicine(@RequestBody Medicine var1) {  
 throw new Error("Unresolved compilation problems: \n\tPostMapping cannot be resolved to a type\n\tRequestBody cannot be resolved to a type\n\tThe method save(Medicine) is undefined for the type MedicineRepository\n");  
 }  
  
 public ResponseEntity<Medicine> getPatientById(@PathVariable Long var1) throws AttributeNotFoundException {  
 throw new Error("Unresolved compilation problems: \n\tGetMapping cannot be resolved to a type\n\tResponseEntity cannot be resolved to a type\n\tPathVariable cannot be resolved to a type\n\tThe method findById(Long) is undefined for the type MedicineRepository\n\tResponseEntity cannot be resolved\n");  
 }  
  
 public ResponseEntity<Medicine> updateMedicine(@PathVariable Long var1, @RequestBody Medicine var2) throws AttributeNotFoundException {  
 throw new Error("Unresolved compilation problems: \n\tPutMapping cannot be resolved to a type\n\tResponseEntity cannot be resolved to a type\n\tPathVariable cannot be resolved to a type\n\tRequestBody cannot be resolved to a type\n\tThe method findById(Long) is undefined for the type MedicineRepository\n\tThe method save(Medicine) is undefined for the type MedicineRepository\n\tResponseEntity cannot be resolved\n");  
 }  
  
 public ResponseEntity<Map<String, Boolean>> deleteMedicine(@PathVariable Long var1) throws AttributeNotFoundException {  
 throw new Error("Unresolved compilation problems: \n\tDeleteMapping cannot be resolved to a type\n\tResponseEntity cannot be resolved to a type\n\tPathVariable cannot be resolved to a type\n\tThe method findById(Long) is undefined for the type MedicineRepository\n\tThe method delete(Medicine) is undefined for the type MedicineRepository\n\tResponseEntity cannot be resolved\n");  
 }  
}

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.example.doclogin;  
  
@SpringBootApplication  
public class DocLoginApplication {  
 public DocLoginApplication() {  
 throw new Error("Unresolved compilation problems: \n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tSpringBootApplication cannot be resolved to a type\n\tSpringApplication cannot be resolved\n");  
 }  
  
 public static void main(String[] var0) {  
 throw new Error("Unresolved compilation problem: \n\tSpringApplication cannot be resolved\n");  
 }  
}

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.example.doclogin;  
  
@SpringBootTest  
class AppointmentApplicationTests {  
 AppointmentApplicationTests() {  
 throw new Error("Unresolved compilation problems: \n\tThe import org.junit cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tSpringBootTest cannot be resolved to a type\n\tTest cannot be resolved to a type\n");  
 }  
  
 @Test  
 void contextLoads() {  
 throw new Error("Unresolved compilation problem: \n\tTest cannot be resolved to a type\n");  
 }  
}

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.example.doclogin;  
  
@SpringBootApplication  
public class AppointmentApplication {  
 public AppointmentApplication() {  
 throw new Error("Unresolved compilation problems: \n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tSpringBootApplication cannot be resolved to a type\n\tSpringApplication cannot be resolved\n");  
 }  
  
 public static void main(String[] var0) {  
 throw new Error("Unresolved compilation problem: \n\tSpringApplication cannot be resolved\n");  
 }  
}

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.example.doclogin.repository;  
  
@Repository  
public interface MedicineRepository {  
}

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.example.doclogin.repository;  
  
@Repository  
public interface AppointmentRepository {  
}

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.example.doclogin.model;  
  
@Entity  
@Table  
public class Medicine {  
 @Id  
 @GeneratedValue  
 private long id;  
 @Column  
 private String drugName;  
 @Column  
 private String stock;  
  
 public Medicine() {  
 throw new Error("Unresolved compilation problems: \n\tThe import javax.persistence cannot be resolved\n\tThe import javax.persistence cannot be resolved\n\tThe import javax.persistence cannot be resolved\n\tThe import javax.persistence cannot be resolved\n\tThe import javax.persistence cannot be resolved\n\tThe import javax.persistence cannot be resolved\n\tEntity cannot be resolved to a type\n\tTable cannot be resolved to a type\n\tId cannot be resolved to a type\n\tGeneratedValue cannot be resolved to a type\n\tGenerationType cannot be resolved to a variable\n\tColumn cannot be resolved to a type\n\tColumn cannot be resolved to a type\n");  
 }  
  
 public Medicine(String var1, String var2) {  
 throw new Error("Unresolved compilation problems: \n\tThe import javax.persistence cannot be resolved\n\tThe import javax.persistence cannot be resolved\n\tThe import javax.persistence cannot be resolved\n\tThe import javax.persistence cannot be resolved\n\tThe import javax.persistence cannot be resolved\n\tThe import javax.persistence cannot be resolved\n\tEntity cannot be resolved to a type\n\tTable cannot be resolved to a type\n\tId cannot be resolved to a type\n\tGeneratedValue cannot be resolved to a type\n\tGenerationType cannot be resolved to a variable\n\tColumn cannot be resolved to a type\n\tColumn cannot be resolved to a type\n");  
 }  
  
 public long getId() {  
 throw new Error("Unresolved compilation problem: \n");  
 }  
  
 public void setId(long var1) {  
 throw new Error("Unresolved compilation problem: \n");  
 }  
  
 public String getDrugName() {  
 throw new Error("Unresolved compilation problem: \n");  
 }  
  
 public void setDrugName(String var1) {  
 throw new Error("Unresolved compilation problem: \n");  
 }  
  
 public String getStock() {  
 throw new Error("Unresolved compilation problem: \n");  
 }  
  
 public void setStock(String var1) {  
 throw new Error("Unresolved compilation problem: \n");  
 }  
}

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.example.doclogin.model;  
  
@Entity  
@Table  
public class Appointment {  
 @Id  
 private long id;  
 @GeneratedValue  
 @Column  
 private String name;  
 @Column  
 private String age;  
 @Column  
 private String symptoms;  
 @Column  
 private String number;  
  
 public Appointment() {  
 throw new Error("Unresolved compilation problems: \n\tThe import javax.persistence cannot be resolved\n\tThe import javax.persistence cannot be resolved\n\tThe import javax.persistence cannot be resolved\n\tThe import javax.persistence cannot be resolved\n\tThe import javax.persistence cannot be resolved\n\tThe import javax.persistence cannot be resolved\n\tEntity cannot be resolved to a type\n\tTable cannot be resolved to a type\n\tId cannot be resolved to a type\n\tGeneratedValue cannot be resolved to a type\n\tGenerationType cannot be resolved to a variable\n\tColumn cannot be resolved to a type\n\tColumn cannot be resolved to a type\n\tColumn cannot be resolved to a type\n\tColumn cannot be resolved to a type\n");  
 }  
  
 public Appointment(String var1, String var2, String var3, String var4) {  
 throw new Error("Unresolved compilation problems: \n\tThe import javax.persistence cannot be resolved\n\tThe import javax.persistence cannot be resolved\n\tThe import javax.persistence cannot be resolved\n\tThe import javax.persistence cannot be resolved\n\tThe import javax.persistence cannot be resolved\n\tThe import javax.persistence cannot be resolved\n\tEntity cannot be resolved to a type\n\tTable cannot be resolved to a type\n\tId cannot be resolved to a type\n\tGeneratedValue cannot be resolved to a type\n\tGenerationType cannot be resolved to a variable\n\tColumn cannot be resolved to a type\n\tColumn cannot be resolved to a type\n\tColumn cannot be resolved to a type\n\tColumn cannot be resolved to a type\n");  
 }  
  
 public long getId() {  
 throw new Error("Unresolved compilation problem: \n");  
 }  
  
 public void setId(long var1) {  
 throw new Error("Unresolved compilation problem: \n");  
 }  
  
 public String getName() {  
 throw new Error("Unresolved compilation problem: \n");  
 }  
  
 public void setName(String var1) {  
 throw new Error("Unresolved compilation problem: \n");  
 }  
  
 public String getAge() {  
 throw new Error("Unresolved compilation problem: \n");  
 }  
  
 public void setAge(String var1) {  
 throw new Error("Unresolved compilation problem: \n");  
 }  
  
 public String getSymptoms() {  
 throw new Error("Unresolved compilation problem: \n");  
 }  
  
 public void setSymptoms(String var1) {  
 throw new Error("Unresolved compilation problem: \n");  
 }  
  
 public String getNumber() {  
 throw new Error("Unresolved compilation problem: \n");  
 }  
  
 public void setNumber(String var1) {  
 throw new Error("Unresolved compilation problem: \n");  
 }  
}

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.example.hospital;  
  
@SpringBootTest  
class DemoApplicationTests {  
 DemoApplicationTests() {  
 throw new Error("Unresolved compilation problems: \n\tThe import org.junit cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tSpringBootTest cannot be resolved to a type\n\tTest cannot be resolved to a type\n");  
 }  
  
 @Test  
 void contextLoads() {  
 throw new Error("Unresolved compilation problem: \n\tTest cannot be resolved to a type\n");  
 }  
}

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.example.hospital;  
  
@SpringBootApplication  
public class DemoApplication {  
 public DemoApplication() {  
 throw new Error("Unresolved compilation problems: \n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tSpringBootApplication cannot be resolved to a type\n\tSpringApplication cannot be resolved\n");  
 }  
  
 public static void main(String[] var0) {  
 throw new Error("Unresolved compilation problem: \n\tSpringApplication cannot be resolved\n");  
 }  
}

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.example.hospital.service;  
  
import com.example.hospital.model.patient;  
import com.example.hospital.repository.PatientRepository;  
  
public class service {  
 @Autowired  
 private PatientRepository patientRepository;  
  
 public service() {  
 throw new Error("Unresolved compilation problems: \n\tThe import org.springframework cannot be resolved\n\tAutowired cannot be resolved to a type\n\tThe method findById(Long) is undefined for the type PatientRepository\n\tThe method save(patient) is undefined for the type PatientRepository\n");  
 }  
  
 public patient updatePatient(patient var1) {  
 throw new Error("Unresolved compilation problems: \n\tThe method findById(Long) is undefined for the type PatientRepository\n\tThe method save(patient) is undefined for the type PatientRepository\n");  
 }  
}

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.example.hospital.repository;  
  
@Repository  
public interface PatientRepository {  
 static {  
 throw new Error("Unresolved compilation problems: \n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tRepository cannot be resolved to a type\n\tJpaRepository cannot be resolved to a type\n");  
 }  
  
 void deleteById(Long var1);  
}

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.example.hospital.model;  
  
@Entity  
@Table  
public class patient {  
 @Id  
 @GeneratedValue  
 private long id;  
 @Column  
 private String name;  
 @Column  
 private String age;  
 @Column  
 private String blood;  
 @Column  
 private String prescription;  
 @Column  
 private String dose;  
 @Column  
 private String fees;  
 @Column  
 private String urgency;  
  
 public patient() {  
 throw new Error("Unresolved compilation problems: \n\tThe import javax.persistence cannot be resolved\n\tThe import javax.persistence cannot be resolved\n\tThe import javax.persistence cannot be resolved\n\tThe import javax.persistence cannot be resolved\n\tThe import javax.persistence cannot be resolved\n\tThe import javax.persistence cannot be resolved\n\tEntity cannot be resolved to a type\n\tTable cannot be resolved to a type\n\tId cannot be resolved to a type\n\tGeneratedValue cannot be resolved to a type\n\tGenerationType cannot be resolved to a variable\n\tColumn cannot be resolved to a type\n\tColumn cannot be resolved to a type\n\tColumn cannot be resolved to a type\n\tColumn cannot be resolved to a type\n\tColumn cannot be resolved to a type\n\tColumn cannot be resolved to a type\n\tColumn cannot be resolved to a type\n");  
 }  
  
 public patient(String var1, String var2, String var3, String var4, String var5, String var6, String var7) {  
 throw new Error("Unresolved compilation problems: \n\tThe import javax.persistence cannot be resolved\n\tThe import javax.persistence cannot be resolved\n\tThe import javax.persistence cannot be resolved\n\tThe import javax.persistence cannot be resolved\n\tThe import javax.persistence cannot be resolved\n\tThe import javax.persistence cannot be resolved\n\tEntity cannot be resolved to a type\n\tTable cannot be resolved to a type\n\tId cannot be resolved to a type\n\tGeneratedValue cannot be resolved to a type\n\tGenerationType cannot be resolved to a variable\n\tColumn cannot be resolved to a type\n\tColumn cannot be resolved to a type\n\tColumn cannot be resolved to a type\n\tColumn cannot be resolved to a type\n\tColumn cannot be resolved to a type\n\tColumn cannot be resolved to a type\n\tColumn cannot be resolved to a type\n");  
 }  
  
 public String getName() {  
 throw new Error("Unresolved compilation problem: \n");  
 }  
  
 public long getId() {  
 throw new Error("Unresolved compilation problem: \n");  
 }  
  
 public void setId(long var1) {  
 throw new Error("Unresolved compilation problem: \n");  
 }  
  
 public void setName(String var1) {  
 throw new Error("Unresolved compilation problem: \n");  
 }  
  
 public String getAge() {  
 throw new Error("Unresolved compilation problem: \n");  
 }  
  
 public void setAge(String var1) {  
 throw new Error("Unresolved compilation problem: \n");  
 }  
  
 public String getBlood() {  
 throw new Error("Unresolved compilation problem: \n");  
 }  
  
 public void setBlood(String var1) {  
 throw new Error("Unresolved compilation problem: \n");  
 }  
  
 public String getPrescription() {  
 throw new Error("Unresolved compilation problem: \n");  
 }  
  
 public void setPrescription(String var1) {  
 throw new Error("Unresolved compilation problem: \n");  
 }  
  
 public String getDose() {  
 throw new Error("Unresolved compilation problem: \n");  
 }  
  
 public void setDose(String var1) {  
 throw new Error("Unresolved compilation problem: \n");  
 }  
  
 public String getFees() {  
 throw new Error("Unresolved compilation problem: \n");  
 }  
  
 public void setFees(String var1) {  
 throw new Error("Unresolved compilation problem: \n");  
 }  
  
 public String getUrgency() {  
 throw new Error("Unresolved compilation problem: \n");  
 }  
  
 public void setUrgency(String var1) {  
 throw new Error("Unresolved compilation problem: \n");  
 }  
}

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.example.hospital.controller;  
  
import com.example.hospital.model.patient;  
import com.example.hospital.repository.PatientRepository;  
import java.util.List;  
import java.util.Map;  
import javax.management.AttributeNotFoundException;  
  
@RestController  
@CrossOrigin  
public class PatientController {  
 @Autowired  
 private PatientRepository patientRepository;  
  
 public PatientController() {  
 throw new Error("Unresolved compilation problems: \n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tThe import org.springframework cannot be resolved\n\tRestController cannot be resolved to a type\n\tCrossOrigin cannot be resolved to a type\n\tRequestMapping cannot be resolved to a type\n\tAutowired cannot be resolved to a type\n\tCrossOrigin cannot be resolved to a type\n\tGetMapping cannot be resolved to a type\n\tThe method findAll() is undefined for the type PatientRepository\n\tCrossOrigin cannot be resolved to a type\n\tPostMapping cannot be resolved to a type\n\tRequestBody cannot be resolved to a type\n\tThe method save(patient) is undefined for the type PatientRepository\n\tGetMapping cannot be resolved to a type\n\tResponseEntity cannot be resolved to a type\n\tPathVariable cannot be resolved to a type\n\tThe method findById(Long) is undefined for the type PatientRepository\n\tResponseEntity cannot be resolved\n\tPutMapping cannot be resolved to a type\n\tResponseEntity cannot be resolved to a type\n\tPathVariable cannot be resolved to a type\n\tRequestBody cannot be resolved to a type\n\tThe method findById(Long) is undefined for the type PatientRepository\n\tThe method save(patient) is undefined for the type PatientRepository\n\tResponseEntity cannot be resolved\n\tDeleteMapping cannot be resolved to a type\n\tResponseEntity cannot be resolved to a type\n\tPathVariable cannot be resolved to a type\n\tThe method findById(Long) is undefined for the type PatientRepository\n\tThe method delete(patient) is undefined for the type PatientRepository\n\tResponseEntity cannot be resolved\n");  
 }  
  
 public @CrossOrigin List<patient> getAllPatients() {  
 throw new Error("Unresolved compilation problems: \n\tCrossOrigin cannot be resolved to a type\n\tGetMapping cannot be resolved to a type\n\tThe method findAll() is undefined for the type PatientRepository\n");  
 }  
  
 public @CrossOrigin patient createPatient(@RequestBody patient var1) {  
 throw new Error("Unresolved compilation problems: \n\tCrossOrigin cannot be resolved to a type\n\tPostMapping cannot be resolved to a type\n\tRequestBody cannot be resolved to a type\n\tThe method save(patient) is undefined for the type PatientRepository\n");  
 }  
  
 public ResponseEntity<patient> getPatientById(@PathVariable Long var1) throws AttributeNotFoundException {  
 throw new Error("Unresolved compilation problems: \n\tGetMapping cannot be resolved to a type\n\tResponseEntity cannot be resolved to a type\n\tPathVariable cannot be resolved to a type\n\tThe method findById(Long) is undefined for the type PatientRepository\n\tResponseEntity cannot be resolved\n");  
 }  
  
 public ResponseEntity<patient> updatePatient(@PathVariable Long var1, @RequestBody patient var2) throws AttributeNotFoundException {  
 throw new Error("Unresolved compilation problems: \n\tPutMapping cannot be resolved to a type\n\tResponseEntity cannot be resolved to a type\n\tPathVariable cannot be resolved to a type\n\tRequestBody cannot be resolved to a type\n\tThe method findById(Long) is undefined for the type PatientRepository\n\tThe method save(patient) is undefined for the type PatientRepository\n\tResponseEntity cannot be resolved\n");  
 }  
  
 public ResponseEntity<Map<String, Boolean>> deletePatient(@PathVariable Long var1) throws AttributeNotFoundException {  
 throw new Error("Unresolved compilation problems: \n\tDeleteMapping cannot be resolved to a type\n\tResponseEntity cannot be resolved to a type\n\tPathVariable cannot be resolved to a type\n\tThe method findById(Long) is undefined for the type PatientRepository\n\tThe method delete(patient) is undefined for the type PatientRepository\n\tResponseEntity cannot be resolved\n");  
 }  
}

// This file is required by karma.conf.js and loads recursively all the .spec and framework files

import 'zone.js/testing';

import { getTestBed } from '@angular/core/testing';

import {

BrowserDynamicTestingModule,

platformBrowserDynamicTesting

} from '@angular/platform-browser-dynamic/testing';

declare const require: {

context(path: string, deep?: boolean, filter?: RegExp): {

<T>(id: string): T;

keys(): string[];

};

};

// First, initialize the Angular testing environment.

getTestBed().initTestEnvironment(

BrowserDynamicTestingModule,

platformBrowserDynamicTesting(),

);

// Then we find all the tests.

const context = require.context('./', true, /\.spec\.ts$/);

// And load the modules.

context.keys().forEach(context);

/\* You can add global styles to this file, and also import other style files \*/

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

/\*\*

\* This file includes polyfills needed by Angular and is loaded before the app.

\* You can add your own extra polyfills to this file.

\*

\* This file is divided into 2 sections:

\* 1. Browser polyfills. These are applied before loading ZoneJS and are sorted by browsers.

\* 2. Application imports. Files imported after ZoneJS that should be loaded before your main

\* file.

\*

\* The current setup is for so-called "evergreen" browsers; the last versions of browsers that

\* automatically update themselves. This includes recent versions of Safari, Chrome (including

\* Opera), Edge on the desktop, and iOS and Chrome on mobile.

\*

\* Learn more in <https://angular.io/guide/browser-support>

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* BROWSER POLYFILLS

\*/

/\*\*

\* By default, zone.js will patch all possible macroTask and DomEvents

\* user can disable parts of macroTask/DomEvents patch by setting following flags

\* because those flags need to be set before `zone.js` being loaded, and webpack

\* will put import in the top of bundle, so user need to create a separate file

\* in this directory (for example: zone-flags.ts), and put the following flags

\* into that file, and then add the following code before importing zone.js.

\* import './zone-flags';

\*

\* The flags allowed in zone-flags.ts are listed here.

\*

\* The following flags will work for all browsers.

\*

\* (window as any).\_\_Zone\_disable\_requestAnimationFrame = true; // disable patch requestAnimationFrame

\* (window as any).\_\_Zone\_disable\_on\_property = true; // disable patch onProperty such as onclick

\* (window as any).\_\_zone\_symbol\_\_UNPATCHED\_EVENTS = ['scroll', 'mousemove']; // disable patch specified eventNames

\*

\* in IE/Edge developer tools, the addEventListener will also be wrapped by zone.js

\* with the following flag, it will bypass `zone.js` patch for IE/Edge

\*

\* (window as any).\_\_Zone\_enable\_cross\_context\_check = true;

\*

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Zone JS is required by default for Angular itself.

\*/

import 'zone.js'; // Included with Angular CLI.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* APPLICATION IMPORTS

\*/

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

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

import { AppModule } from './app/app.module';

import { environment } from './environments/environment';

if (environment.production) {

enableProdMode();

}

platformBrowserDynamic().bootstrapModule(AppModule)

.catch(err => console.error(err));

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>HospitalProject</title>

<base href="/">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="icon" type="image/x-icon" href="favicon.ico">

</head>

<body>

<app-root></app-root>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"

integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin="anonymous">

<script src="https://code.jquery.com/jquery-3.4.1.slim.min.js"

integrity="sha384-J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n"

crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"

integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"

crossorigin="anonymous"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"

integrity="sha384-wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6"

crossorigin="anonymous"></script>

</body>

</html>

export const environment = {

production: true

};

// This file can be replaced during build by using the `fileReplacements` array.

// `ng build` replaces `environment.ts` with `environment.prod.ts`.

// The list of file replacements can be found in `angular.json`.

export const environment = {

production: false

};

/\*

\* For easier debugging in development mode, you can import the following file

\* to ignore zone related error stack frames such as `zone.run`, `zoneDelegate.invokeTask`.

\*

\* This import should be commented out in production mode because it will have a negative impact

\* on performance if an error is thrown.

\*/

// import 'zone.js/plugins/zone-error'; // Included with Angular CLI.

import { TestBed } from '@angular/core/testing';

import { AdminauthService } from './adminauth.service';

describe('AdminauthService', () => {

let service: AdminauthService;

beforeEach(() => {

TestBed.configureTestingModule({});

service = TestBed.inject(AdminauthService);

});

it('should be created', () => {

expect(service).toBeTruthy();

});

});

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

import { Router } from '@angular/router';

@Injectable({

providedIn: 'root'

})

export class AdminauthService {

constructor() { }

authenticate(username2: string, password2: string) {

if (username2 === "Rishi" && password2 === "Prem@13") {

sessionStorage.setItem('username2', username2)

return true;

} else {

return false;

}

}

isUserLoggedIn() {

let user = sessionStorage.getItem('username2')

console.log(!(user === null))

return !(user === null)

}

logOut() {

sessionStorage.removeItem('username2')

}

}

import { TestBed } from '@angular/core/testing';

import { AdminauthguardService } from './adminauthguard.service';

describe('AdminauthguardService', () => {

let service: AdminauthguardService;

beforeEach(() => {

TestBed.configureTestingModule({});

service = TestBed.inject(AdminauthguardService);

});

it('should be created', () => {

expect(service).toBeTruthy();

});

});

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

import { CanActivate } from '@angular/router';

import { ActivatedRouteSnapshot, CanActivate, Router, RouterStateSnapshot } from '@angular/router';

import { Observable } from 'rxjs';

import { AdminauthService } from './adminauth.service';

@Injectable({

providedIn: 'root'

})

export class AdminauthguardService implements CanActivate {

constructor(private router: Router, private authService: AdminauthService) { }

canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot

): boolean | Promise<boolean> | Observable<boolean>{

if (this.authService.isUserLoggedIn())

{

return true;

}

else{

this.router.navigate(['home']);

return false;

}

}

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

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = 'Hospital\_project';

}

import { TestBed } from '@angular/core/testing';

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

import { RouterTestingModule } from '@angular/router/testing';

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

describe('AppComponent', () => {

beforeEach(async () => {

await TestBed.configureTestingModule({

imports: [

RouterTestingModule,

FormsModule

],

declarations: [

AppComponent

],

}).compileComponents();

});

it('should create the app', () => {

const fixture = TestBed.createComponent(AppComponent);

const app = fixture.componentInstance;

expect(app).toBeTruthy();

});

it(`should have as title 'Hospital\_project'`, () => {

const fixture = TestBed.createComponent(AppComponent);

const app = fixture.componentInstance;

expect(app.title).toEqual('Hospital\_project');

});

it('should render title', () => {

const fixture = TestBed.createComponent(AppComponent);

fixture.detectChanges();

const compiled = fixture.nativeElement as HTMLElement;

expect(compiled.querySelector('.content span')?.textContent).toContain('Hospital\_project app is running!');

});

});

<router-outlet></router-outlet>

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

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

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

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

import { NewsfeedComponent } from './newsfeed/newsfeed.component';

import { DocloginComponent } from './doclogin/doclogin.component';

import { AdminloginComponent } from './adminlogin/adminlogin.component';

import { DocdashComponent } from './docdash/docdash.component';

import { AdmindashComponent } from './admindash/admindash.component';

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

import { Ng2SearchPipeModule } from 'ng2-search-filter';

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

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

import { AuthGaurdService } from './auth-gaurd.service';

import { CreatepatientComponent } from './createpatient/createpatient.component';

import { AuthenticationService } from './authentication.service';

import { UpdatePatientComponent } from './update-patient/update-patient.component';

import { MedicineListComponent } from './medicine-list/medicine-list.component';

import { CreatemedicineComponent } from './createmedicine/createmedicine.component';

import { UpdateMedicineComponent } from './update-medicine/update-medicine.component';

import { AppointmentListComponent } from './appointment-list/appointment-list.component';

import { CreateAppointmentComponent } from './create-appointment/create-appointment.component';

import { ViewPatientComponent } from './view-patient/view-patient.component';

const routes: Routes = [

{ path: '', component: NewsfeedComponent },

{ path: 'doclogin', component: DocloginComponent },

{ path: 'adlogin', component: AdminloginComponent },

{ path: 'home', component: NewsfeedComponent },

{ path: 'createpatient', component: CreatepatientComponent, canActivate: [AuthGaurdService] },

{ path: 'docdash', component: DocdashComponent, canActivate: [AuthGaurdService] },

{ path: 'updatepatient/:id', component: UpdatePatientComponent, canActivate: [AuthGaurdService] },

{ path: 'admindash', component: AdmindashComponent, canActivate: [AuthGaurdService] },

{ path: 'medicinelist', component: MedicineListComponent, canActivate: [AuthGaurdService] },

{ path: 'createmedicine', component: CreatemedicineComponent, canActivate: [AuthGaurdService] },

{ path: 'updatemedicine/:id', component: UpdateMedicineComponent, canActivate: [AuthGaurdService] },

{ path: 'appointmentlist', component: AppointmentListComponent, canActivate: [AuthGaurdService] },

{ path: 'createappointment', component: CreateAppointmentComponent, canActivate: [AuthGaurdService] },

{ path: 'viewpatient/:id', component: ViewPatientComponent }

]

@NgModule({

declarations: [

AppComponent,

NewsfeedComponent,

DocloginComponent,

AdminloginComponent,

DocdashComponent,

AdmindashComponent,

CreatepatientComponent,

UpdatePatientComponent,

MedicineListComponent,

CreatemedicineComponent,

UpdateMedicineComponent,

AppointmentListComponent,

CreateAppointmentComponent,

ViewPatientComponent

],

imports: [

RouterModule.forRoot(routes),

FormsModule,

BrowserModule,

Ng2SearchPipeModule,

AppRoutingModule,

HttpClientModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

import { TestBed } from '@angular/core/testing';

import { AppointmentService } from './appointment.service';

describe('AppointmentService', () => {

let service: AppointmentService;

beforeEach(() => {

TestBed.configureTestingModule({});

service = TestBed.inject(AppointmentService);

});

it('should be created', () => {

expect(service).toBeTruthy();

});

});

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

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

const routes: Routes = [];

@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule]

})

export class AppRoutingModule { }

import { Appointment } from './appointment';

describe('Appointment', () => {

it('should create an instance', () => {

expect(new Appointment()).toBeTruthy();

});

});

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

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

import { Observable } from 'rxjs';

import { Appointment } from './appointment';

@Injectable({

providedIn: 'root'

})

export class AppointmentService {

private baseUrl = "<http://localhost:8899/api/v3/appointments>";

constructor(private httpClient: HttpClient) { }

getAppointmentslist(): Observable<Appointment[]> {

return this.httpClient.get<Appointment[]>(`${this.baseUrl}`);

}

createAppointment(appointment: Appointment): Observable<Appointment> {

return this.httpClient.post<Appointment>(`${this.baseUrl}`, appointment);

}

getAppointmentById(id: number): Observable<Appointment> {

return this.httpClient.get<Appointment>(`${this.baseUrl}/${id}`);

}

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

return this.httpClient.delete(`${this.baseUrl}/${id}`);

}

}

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

import { ActivatedRouteSnapshot, CanActivate, Router, RouterStateSnapshot } from '@angular/router';

import { Observable } from 'rxjs';

import { AuthenticationService } from './authentication.service';

@Injectable({

providedIn: 'root'

})

export class AuthGaurdService implements CanActivate {

constructor(private router: Router, private authService: AuthenticationService) { }

canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot

): boolean | Promise<boolean> | Observable<boolean>{

if (this.authService.isUserLoggedIn())

{

return true;

}

else{

this.router.navigate(['home']);

return false;

}

}

}

import { TestBed } from '@angular/core/testing';

import { AuthGaurdService } from './auth-gaurd.service';

describe('AuthGaurdService', () => {

let service: AuthGaurdService;

beforeEach(() => {

TestBed.configureTestingModule({});

service = TestBed.inject(AuthGaurdService);

});

it('should be created', () => {

expect(service).toBeTruthy();

});

});

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

import { Router } from '@angular/router';

@Injectable({

providedIn: 'root'

})

export class AuthenticationService {

constructor() { }

authenticate(username: string, password: string) {

if (username === "Rishi" && password === "Prem@12") {

sessionStorage.setItem('username', username)

return true;

} else {

return false;

}

}

isUserLoggedIn() {

let user = sessionStorage.getItem('username')

console.log(!(user === null))

return !(user === null)

}

logOut() {

sessionStorage.removeItem('username')

}

}

import { TestBed } from '@angular/core/testing';

import { AuthenticationService } from './authentication.service';

describe('AuthenticationService', () => {

let service: AuthenticationService;

beforeEach(() => {

TestBed.configureTestingModule({});

service = TestBed.inject(AuthenticationService);

});

it('should be created', () => {

expect(service).toBeTruthy();

});

});

export class Medicine {

id: number;

drugName: string;

stock: string;

}

import { Medicine } from './medicine';

describe('Medicine', () => {

it('should create an instance', () => {

expect(new Medicine()).toBeTruthy();

});

});

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

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

import { Observable } from 'rxjs';

import { Medicine } from './medicine';

@Injectable({

providedIn: 'root'

})

export class MedicineService {

private baseUrl = '<http://localhost:8090/api/v2/medicines>';

constructor(private httpClient: HttpClient) { }

getMedicinesList(): Observable<Medicine[]>{

return this.httpClient.get<Medicine[]>(`${this.baseUrl}`);

}

createMedicine(medicine: Medicine): Observable<Medicine> {

return this.httpClient.post<Medicine>(`${this.baseUrl}`, medicine);

}

getMedicineById(id: number): Observable<Medicine> {

return this.httpClient.get<Medicine>(`${this.baseUrl}/${id}`);

}

updateMedicine(id: number, medicine: Medicine): Observable<Object> {

return this.httpClient.put(`${this.baseUrl}/${id}`, medicine);

}

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

return this.httpClient.delete(`${this.baseUrl}/${id}`);

}

}

import { TestBed } from '@angular/core/testing';

import { MedicineService } from './medicine.service';

describe('MedicineService', () => {

let service: MedicineService;

beforeEach(() => {

TestBed.configureTestingModule({});

service = TestBed.inject(MedicineService);

});

it('should be created', () => {

expect(service).toBeTruthy();

});

});

export class Patient {

id: number;

name: string;

age: string;

blood: string;

prescription: string;

dose: string;

fees: string;

urgency: string;

}

import { Patient } from './patient';

describe('Patient', () => {

it('should create an instance', () => {

expect(new Patient()).toBeTruthy();

});

});

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

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

import { Observable } from 'rxjs';

import { Patient } from './patient';

@Injectable({

providedIn: 'root'

})

export class PatientService {

private baseUrl = "<http://localhost:8080/api/v1/patients>";

constructor(private httpClient: HttpClient) { }

getPatientslist(): Observable<Patient[]> {

return this.httpClient.get<Patient[]>(`${this.baseUrl}`);

}

createPatient(patient: Patient): Observable<Patient> {

return this.httpClient.post<Patient>(`${this.baseUrl}`, patient);

}

getPatientById(id: number): Observable<Patient> {

return this.httpClient.get<Patient>(`${this.baseUrl}/${id}`);

}

updatePatient(id: number, patient: Patient): Observable<Object> {

return this.httpClient.put(`${this.baseUrl}/${id}`, patient);

}

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

return this.httpClient.delete(`${this.baseUrl}/${id}`);

}

}

import { TestBed } from '@angular/core/testing';

import { PatientService } from './patient.service';

describe('PatientService', () => {

let service: PatientService;

beforeEach(() => {

TestBed.configureTestingModule({});

service = TestBed.inject(PatientService);

});

it('should be created', () => {

expect(service).toBeTruthy();

});

});

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

import { Router } from '@angular/router';

import { Patient } from '../patient';

import { PatientService } from '../patient.service';

@Component({

selector: 'app-admindash',

templateUrl: './admindash.component.html',

styleUrls: ['./admindash.component.css']

})

export class AdmindashComponent implements OnInit {

searchText: string;

patients: Patient[];

constructor(private patientService: PatientService,

private router: Router) { }

ngOnInit(): void {

this.getPatients();

}

private getPatients(){

this.patientService.getPatientslist().subscribe(data => { this.patients = data;

});

}

updatePatient(id: number) {

this.router.navigate(['updatepatient', id]);

}

deletePatient(id: number) {

this.patientService.deletePatient(id).subscribe(data => {

console.log(data);

this.getPatients();

} );

}

}

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { AdmindashComponent } from './admindash.component';

describe('AdmindashComponent', () => {

let component: AdmindashComponent;

let fixture: ComponentFixture<AdmindashComponent>;

beforeEach(async () => {

await TestBed.configureTestingModule({

declarations: [ AdmindashComponent ]

})

.compileComponents();

fixture = TestBed.createComponent(AdmindashComponent);

component = fixture.componentInstance;

fixture.detectChanges();

});

it('should create', () => {

expect(component).toBeTruthy();

});

});

<nav style="margin:auto" class="navbar navbar-expand-lg bg-dark">

<a style="color: white;" class="navbar-brand" href="#">APOLLO HOSPITALS</a>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav me-auto mb-2 mb-lg-0">

<li class="nav-item">

<a style="color: white;"class="nav-link active" aria-current="page" href="javascript:void(0);" routerLink="/appointmentlist">Appointments</a>

</li>

</ul>

<div class="mx-2">

<button style="margin: 5px;"class="btn btn-danger" data-toggle="modal" data-target="#loginModal"><a style="color:white;" href="javascript:void(0);" routerLink="/home">Logout</a></button>

</div>

</div>

</nav>

<div class="container">

<hr>

<hr>

<hr>

<h2>Admin Dashboard</h2>

<a style="color: rgb(86, 86, 86); font-size: x-large;"><marquee behavior="scroll" direction="left">

Welcome to Admin Dashboard. Please find the patient list below!!. Click Appointments to view and update Appointments, Click Add Appointment to add new Appointment</marquee></a>

<hr>

<input type="text" name="search" [(ngModel)]="searchText" placeholder="Search">

<br>

<a style="color: rgb(86, 86, 86); font-size: xx-large;">Current Patient List in the Queue</a>

<div class="tableFixHead">

<br>

<table class="table table-striped">

<thead class="table-header">

<tr>

<th>Id</th>

<th>Name</th>

<th>Age</th>

<th>Fees</th>

<th>Urgency</th>

<th>Action</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let patient of patients | filter:searchText">

<td>{{patient.id}}</td>

<td>{{patient.name}}</td>

<td>{{patient.age}}</td>

<td>{{patient.fees}}</td>

<td>{{patient.urgency}}</td>

<td>

<button (click) = "deletePatient(patient.id)" class="btn btn-danger" style="margin-left: 10px;">Delete</button>

</td>

</tr>

</tbody>

</table>

</div>

.table

{

box-shadow: 10px -10px 5px #CCC;

}

.table-header {

background-color: #95A5A6;

font-size: 14px;

text-transform: uppercase;

letter-spacing: 0.03em;

}

.table-header-row {

background-color: #95A5A6;

font-size: 14px;

text-transform: uppercase;

letter-spacing: 0.03em;

color: #fff;

font-weight: bold;

border-bottom: 1px solid #fff;

}

.navbar{

position: fixed;

top: 0;

width: 100%;

}

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

import { AdminauthService } from '../adminauth.service';

import { Router } from '@angular/router';

@Component({

selector: 'app-adminlogin',

templateUrl: './adminlogin.component.html',

styleUrls: ['./adminlogin.component.css']

})

export class AdminloginComponent implements OnInit {

username2 = 'user'

password2 = ''

invalidLogin = false

constructor(private router:Router, public loginservice: AdminauthService) { }

ngOnInit(): void {

}

checkLogin() {

if (this.loginservice.authenticate(this.username2, this.password2)

) {

this.router.navigate(['admindash'])

this.invalidLogin = false

} else

{

this.invalidLogin = true

alert("Wrong Credentials");

this.router.navigate(['home'])

}

}

}

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { AdminloginComponent } from './adminlogin.component';

describe('AdminloginComponent', () => {

let component: AdminloginComponent;

let fixture: ComponentFixture<AdminloginComponent>;

beforeEach(async () => {

await TestBed.configureTestingModule({

declarations: [ AdminloginComponent ]

})

.compileComponents();

fixture = TestBed.createComponent(AdminloginComponent);

component = fixture.componentInstance;

fixture.detectChanges();

});

it('should create', () => {

expect(component).toBeTruthy();

});

});

<nav style="margin:auto" class="navbar navbar-expand-lg bg-dark">

<a style="color: white;" class="navbar-brand" href="#">APOLLO HOSPITALS</a>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav me-auto mb-2 mb-lg-0">

<li class="nav-item">

<a style="color: white;"class="nav-link active" aria-current="page" href="javascript:void(0);" routerLink="/home">Home</a>

</li>

</ul>

<form class="d-flex" role="search">

<input class="form-control me-2" type="search" placeholder="Search" aria-label="Search">

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

</form>

</div>

</nav>

<div class="container">

<div class="row">

<div class="col-md-6">

<div class="card">

<div class="text-center">

<h1>Login As Admin</h1>

<h6>Please enter your Username & Password</h6>

</div>

<form>

<div class="mb-3">

<label for="formGroupExampleInput" class="form-label"><h4>Username</h4></label>

<input type="text" name="username2" [(ngModel)]="username2" class="form-control" id="formGroupExampleInput" placeholder="Username" required>

</div>

<div class="mb-3">

<label for="formGroupExampleInput2" class="form-label"><h4>Password</h4></label>

<input type="password" name="password2" [(ngModel)]="password2" class="form-control" id="formGroupExampleInput2" placeholder="Password" required>

</div>

<div class="d-grid gap-2 d-md-block">

<button (click)="checkLogin()" style="margin: 5px;" class="btn btn-success" type="button">Login</button>

<button style="margin: 5px;" class="btn btn-warning" type="button" routerLink="/home">cancel</button>

</div>

</form>

</div>

</div>

</div>

</div>

.card{

border: none;

border-radius: 30px;

width: 500px;

padding: 40px;

top: 40%;

left: 30%;

transform: translate(30%, -25%);

background: rgb(158, 219, 219);

justify-content: center;

color: navy;

justify-items: center;

box-shadow: 10px 10px 10px rgb(120, 120, 120);

}

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

import { Router } from '@angular/router';

import { Appointment } from '../appointment';

import { AppointmentService } from '../appointment.service';

@Component({

selector: 'app-appointment-list',

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

styleUrls: ['./appointment-list.component.css']

})

export class AppointmentListComponent implements OnInit {

appointments: Appointment[];

constructor(private appointmentService: AppointmentService,

private router: Router) { }

ngOnInit(): void {

this.getAppointments();

}

private getAppointments() {

this.appointmentService.getAppointmentslist().subscribe(data => {this.appointments = data;

});

}

deleteAppointment(id: number){

this.appointmentService.deleteAppointment(id).subscribe( data => {

console.log(data);

this.getAppointments();

})

}

}

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { AppointmentListComponent } from './appointment-list.component';

describe('AppointmentListComponent', () => {

let component: AppointmentListComponent;

let fixture: ComponentFixture<AppointmentListComponent>;

beforeEach(async () => {

await TestBed.configureTestingModule({

declarations: [ AppointmentListComponent ]

})

.compileComponents();

fixture = TestBed.createComponent(AppointmentListComponent);

component = fixture.componentInstance;

fixture.detectChanges();

});

it('should create', () => {

expect(component).toBeTruthy();

});

});

<nav style="margin:auto" class="navbar navbar-expand-lg bg-dark">

<a style="color: white;" class="navbar-brand" href="#">APOLLO HOSPITALS</a>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav me-auto mb-2 mb-lg-0">

<li class="nav-item">

<a style="color: white;"class="nav-link active" aria-current="page" href="javascript:void(0);" routerLink="/admindash">Back</a>

<li class="nav-item">

<a style="color: white;"class="nav-link active" aria-current="page" href="javascript:void(0);" routerLink="/createappointment">Add Appointments</a>

</ul>

</div>

</nav>

<div class="container">

<hr>

<hr>

<hr>

<h2>Appointment List</h2>

<a style="color: rgb(86, 86, 86); font-size: x-large;">Welcome to Appointment list page. Click Add Appointments to Add appointments</a>

<hr>

<table class="table table-striped">

<thead class="table-header">

<tr>

<th> ID </th>

<th> Name </th>

<th> Age </th>

<th> Symptoms </th>

<th> Number </th>

<th> Action </th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let appointment of appointments">

<td> {{ appointment.id }} </td>

<td> {{ appointment.name }} </td>

<td> {{ appointment.age }} </td>

<td> {{ appointment.symptoms }} </td>

<td> {{ appointment.number }} </td>

<td>

<button (click)="deleteAppointment(appointment.id)" class="btn btn-danger" style="margin-left: 10px"> Delete</button>

</td>

</tr>

</table>

</div>

.table

{

box-shadow: 10px -10px 5px #CCC;

}

.table-header {

background-color: #95A5A6;

font-size: 14px;

text-transform: uppercase;

letter-spacing: 0.03em;

}

.table-header-row {

background-color: #95A5A6;

font-size: 14px;

text-transform: uppercase;

letter-spacing: 0.03em;

color: #fff;

font-weight: bold;

border-bottom: 1px solid #fff;

}

.navbar{

position: fixed;

top: 0;

width: 100%;

}

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

import { Appointment } from '../appointment';

import { AppointmentService } from '../appointment.service';

import { Router } from '@angular/router';

@Component({

selector: 'app-create-appointment',

templateUrl: './create-appointment.component.html',

styleUrls: ['./create-appointment.component.css']

})

export class CreateAppointmentComponent implements OnInit {

appointment: Appointment = new Appointment();

constructor(private appointmentService: AppointmentService,

private router: Router) { }

ngOnInit(): void {

}

saveAppointment() {

this.appointmentService.createAppointment(this.appointment).subscribe(data => {

console.log(data);

this.goToAppointmentList();

},

error => console.log(error));

}

goToAppointmentList() {

this.router.navigate(['/appointmentlist'])

}

onSubmit() {

console.log(this.appointment);

this.saveAppointment();

}

}

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { CreateAppointmentComponent } from './create-appointment.component';

describe('CreateAppointmentComponent', () => {

let component: CreateAppointmentComponent;

let fixture: ComponentFixture<CreateAppointmentComponent>;

beforeEach(async () => {

await TestBed.configureTestingModule({

declarations: [ CreateAppointmentComponent ]

})

.compileComponents();

fixture = TestBed.createComponent(CreateAppointmentComponent);

component = fixture.componentInstance;

fixture.detectChanges();

});

it('should create', () => {

expect(component).toBeTruthy();

});

});

<nav style="margin:auto" class="navbar navbar-expand-lg bg-dark">

<a style="color: white;" class="navbar-brand" href="#">APOLLO HOSPITALS</a>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav me-auto mb-2 mb-lg-0">

<li class="nav-item">

<a style="color: white;"class="nav-link active" aria-current="page" href="javascript:void(0);" routerLink="/appointmentlist">Back</a>

</ul>

</div>

</nav>

<div class="container">

<hr>

<hr>

<hr>

<h2>Add New Appointment</h2>

<form (ngSubmit) = "onSubmit()">

<div class="form-group">

<label>ID</label>

<input type="number" class="form-control" id="id" [(ngModel)]="appointment.id" name="id" placeholder="Enter ID">

</div>

<div class="form-group">

<label>Name</label>

<input type="text" class="form-control" id="name" [(ngModel)]="appointment.name" name="name" placeholder="Enter Name">

</div>

<div class="form-group">

<label>Age</label>

<input type="text" class="form-control" id="age" [(ngModel)]="appointment.age" name="age" placeholder="Enter Age">

</div>

<div class="form-group">

<label>Symptoms</label>

<input type="text" class="form-control" id="symptoms" [(ngModel)]="appointment.symptoms" name="symptoms" placeholder="Enter Symptoms">

</div>

<div class="form-group">

<label>Number</label>

<input type="text" class="form-control" id="number" [(ngModel)]="appointment.number" name="number" placeholder="Enter Number">

</div>

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

</form>

</div>

.table

{

box-shadow: 10px -10px 5px #CCC;

}

.table

{

border-radius:25px

}

.table-header {

background-color: #95A5A6;

font-size: 14px;

text-transform: uppercase;

letter-spacing: 0.03em;

}

.table-header-row {

background-color: #95A5A6;

font-size: 14px;

text-transform: uppercase;

letter-spacing: 0.03em;

color: #fff;

font-weight: bold;

border-bottom: 1px solid #fff;

}

.navbar{

position: fixed;

top: 0;

width: 100%;

}

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

import { Router } from '@angular/router';

import { Medicine } from '../medicine';

import { MedicineService } from '../medicine.service';

@Component({

selector: 'app-createmedicine',

templateUrl: './createmedicine.component.html',

styleUrls: ['./createmedicine.component.css']

})

export class CreatemedicineComponent implements OnInit {

medicine: Medicine = new Medicine();

constructor(private medicineService: MedicineService,

private router: Router) { }

ngOnInit(): void {

}

saveMedicine() {

this.medicineService.createMedicine(this.medicine).subscribe(data => {

console.log(data);

this.goToMedicineList();

},

error => console.log(error));

}

goToMedicineList() {

this.router.navigate(['/medicinelist']);

}

onSubmit() {

console.log(this.medicine);

this.saveMedicine();

}

}

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { CreatemedicineComponent } from './createmedicine.component';

describe('CreatemedicineComponent', () => {

let component: CreatemedicineComponent;

let fixture: ComponentFixture<CreatemedicineComponent>;

beforeEach(async () => {

await TestBed.configureTestingModule({

declarations: [ CreatemedicineComponent ]

})

.compileComponents();

fixture = TestBed.createComponent(CreatemedicineComponent);

component = fixture.componentInstance;

fixture.detectChanges();

});

it('should create', () => {

expect(component).toBeTruthy();

});

});

<nav style="margin:auto" class="navbar navbar-expand-lg bg-dark">

<a style="color: white;" class="navbar-brand" href="#">APOLLO HOSPITALS</a>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav me-auto mb-2 mb-lg-0">

<li class="nav-item">

<a style="color: white;"class="nav-link active" aria-current="page" href="javascript:void(0);" routerLink="/medicinelist">Back</a>

</ul>

</div>

</nav>

<div class="container">

<hr>

<hr>

<hr>

<h3>Add Medicine</h3>

<form (ngSubmit) = "onSubmit()">

<div class="form-group">

<label> ID </label>

<input type="number" class="form-control" [(ngModel)]="medicine.id" id="id" name="id">

</div>

<div class="form-group">

<label> Drug Name</label>

<input type="text" class="form-control" [(ngModel)]="medicine.drugName" id="drugName" name="drugName">

</div>

<div class="form-group">

<label> Stock </label>

<input type="text" class="form-control" [(ngModel)]="medicine.stock" id="stock" name="stock">

</div>

<br>

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

</form>

</div>

.table

{

box-shadow: 10px -10px 5px #CCC;

}

.table

{

border-radius:25px

}

.table-header {

background-color: #95A5A6;

font-size: 14px;

text-transform: uppercase;

letter-spacing: 0.03em;

}

.table-header-row {

background-color: #95A5A6;

font-size: 14px;

text-transform: uppercase;

letter-spacing: 0.03em;

color: #fff;

font-weight: bold;

border-bottom: 1px solid #fff;

}

.navbar{

position: fixed;

top: 0;

width: 100%;

}

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

import { Router } from '@angular/router';

import { Patient } from '../patient';

import { PatientService } from '../patient.service';

@Component({

selector: 'app-createpatient',

templateUrl: './createpatient.component.html',

styleUrls: ['./createpatient.component.css']

})

export class CreatepatientComponent implements OnInit {

patient: Patient = new Patient();

constructor(private patientService: PatientService,

private router: Router) { }

ngOnInit(): void {

}

savePatient() {

this.patientService.createPatient(this.patient).subscribe(data => {

console.log(data);

this.goToPatientList();

},

error => console.log(error));

}

goToPatientList() {

this.router.navigate(['/docdash']);

}

onSubmit() {

console.log(this.patient);

this.savePatient();

}

}

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { CreatepatientComponent } from './createpatient.component';

describe('CreatepatientComponent', () => {

let component: CreatepatientComponent;

let fixture: ComponentFixture<CreatepatientComponent>;

beforeEach(async () => {

await TestBed.configureTestingModule({

declarations: [ CreatepatientComponent ]

})

.compileComponents();

fixture = TestBed.createComponent(CreatepatientComponent);

component = fixture.componentInstance;

fixture.detectChanges();

});

it('should create', () => {

expect(component).toBeTruthy();

});

});

<nav style="margin:auto" class="navbar navbar-expand-lg bg-dark">

<a style="color: white;" class="navbar-brand" href="#">APOLLO HOSPITALS</a>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav me-auto mb-2 mb-lg-0">

<li class="nav-item">

<a style="color: white;"class="nav-link active" aria-current="page" href="javascript:void(0);" routerLink="/docdash">Back</a>

</li>

</ul>

</div>

</nav>

<div class="container">

<hr>

<hr>

<hr>

<div class="col-md-6-offset-md-3">

<h2>Add New Patient</h2>

<form (ngSubmit) = "onSubmit()">

<div class="form-group">

<label>ID</label>

<input type="number" class="form-control" id="id" [(ngModel)]="patient.id" name="id" placeholder="Enter ID">

</div>

<div class="form-group">

<label>Name</label>

<input type="text" class="form-control" name="name" [(ngModel)]="patient.name" name="name" placeholder="Enter Name">

</div>

<div class="form-group">

<label>Age</label>

<input type="text" class="form-control" name="age" [(ngModel)]="patient.age" name="age" placeholder="Enter Age">

</div>

<div class="form-group">

<label>Blood Type</label>

<input type="text" class="form-control" name="blood" [(ngModel)]="patient.blood" name="blood" placeholder="Enter Blood Type">

</div>

<div class="form-group">

<label>Prescription</label>

<input type="text" class="form-control" name="prescription" [(ngModel)]="patient.prescription" name="prescription" placeholder="Enter Prescription">

</div>

<div class="form-group">

<label>Dosage</label>

<input type="text" class="form-control" name="dose" [(ngModel)]="patient.dose" name="dose" placeholder="Enter Dosage">

</div>

<div class="form-group">

<label>Fees</label>

<input type="text" class="form-control" name="fees" [(ngModel)]="patient.fees" name="fees" placeholder="Enter Fees">

</div>

<div class="form-group">

<label>Urgency</label>

<input type="text" class="form-control" name="urgency" [(ngModel)]="patient.urgency" name="urgency" placeholder="Urgency">

</div>

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

</form>

</div>

</div>

.table

{

box-shadow: 10px -10px 5px #CCC;

}

.table

{

border-radius:25px

}

.table-header {

background-color: #95A5A6;

font-size: 14px;

text-transform: uppercase;

letter-spacing: 0.03em;

}

.table-header-row {

background-color: #95A5A6;

font-size: 14px;

text-transform: uppercase;

letter-spacing: 0.03em;

color: #fff;

font-weight: bold;

border-bottom: 1px solid #fff;

}

.navbar{

position: fixed;

top: 0;

width: 100%;

}

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

import { Router } from '@angular/router';

import { Medicine } from '../medicine';

import { MedicineService } from '../medicine.service';

@Component({

selector: 'app-medicine-list',

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

styleUrls: ['./medicine-list.component.css']

})

export class MedicineListComponent implements OnInit {

medicines: Medicine[];

constructor(private medicineService: MedicineService,

private router: Router) { }

ngOnInit(): void {

this.getMedicines();

}

private getMedicines(){

this.medicineService.getMedicinesList().subscribe(data => {this.medicines = data;

});

}

updateMedicine(id: number){

this.router.navigate(['updatemedicine', id]);

}

deleteMedicine(id: number){

this.medicineService.deleteMedicine(id).subscribe( data => {

console.log(data);

this.getMedicines();

})

}

}

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { MedicineListComponent } from './medicine-list.component';

describe('MedicineListComponent', () => {

let component: MedicineListComponent;

let fixture: ComponentFixture<MedicineListComponent>;

beforeEach(async () => {

await TestBed.configureTestingModule({

declarations: [ MedicineListComponent ]

})

.compileComponents();

fixture = TestBed.createComponent(MedicineListComponent);

component = fixture.componentInstance;

fixture.detectChanges();

});

it('should create', () => {

expect(component).toBeTruthy();

});

});

<nav style="margin:auto" class="navbar navbar-expand-lg bg-dark">

<a style="color: white;" class="navbar-brand" href="#">APOLLO HOSPITALS</a>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav me-auto mb-2 mb-lg-0">

<li class="nav-item">

<a style="color: white;"class="nav-link active" aria-current="page" href="javascript:void(0);" routerLink="/docdash">Back</a>

</li>

<li class="nav-item">

<a style="color: white;"class="nav-link active" aria-current="page" href="javascript:void(0);" routerLink="/createmedicine">Add Medicines</a>

</li>

</ul>

</div>

</nav>

<div class="container">

<hr>

<hr>

<hr>

<h2>Medicine List</h2>

<a style="color: rgb(86, 86, 86); font-size: x-large;">Welcome to Medicine list page. Click Add Medicines to Add medicines</a>

<hr>

<table class="table table-striped">

<thead class="table-header">

<tr>

<th> ID </th>

<th> Drug Name </th>

<th> Stock </th>

<th> Action </th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let medicine of medicines">

<td> {{ medicine.id }} </td>

<td> {{ medicine.drugName }} </td>

<td> {{ medicine.stock }} </td>

<td>

<button (click)="updateMedicine(medicine.id)" class="btn btn-info"> Update </button>

<button (click)="deleteMedicine(medicine.id)" class="btn btn-danger" style="margin-left: 10px"> Delete</button>

</td>

</tr>

</table>

</div>

.table

{

box-shadow: 10px -10px 5px #CCC;

}

.table-header {

background-color: #95A5A6;

font-size: 14px;

text-transform: uppercase;

letter-spacing: 0.03em;

}

.table-header-row {

background-color: #95A5A6;

font-size: 14px;

text-transform: uppercase;

letter-spacing: 0.03em;

color: #fff;

font-weight: bold;

border-bottom: 1px solid #fff;

}

.navbar{

position: fixed;

top: 0;

width: 100%;

}

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

import { ActivatedRoute, Router } from '@angular/router';

import { MedicineService } from '../medicine.service';

import { Medicine } from '../medicine';

@Component({

selector: 'app-update-medicine',

templateUrl: './update-medicine.component.html',

styleUrls: ['./update-medicine.component.css']

})

export class UpdateMedicineComponent implements OnInit {

id: number;

medicine: Medicine = new Medicine();

constructor(private medicineService: MedicineService,

private route: ActivatedRoute,

private router: Router) { }

ngOnInit(): void {

this.id = this.route.snapshot.params['id'];

this.medicineService.getMedicineById(this.id).subscribe(data => {

this.medicine = data;

}

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

}

onSubmit() {

this.medicineService.updateMedicine(this.id, this.medicine).subscribe(data => {

this.goToMedicineList();

}

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

}

goToMedicineList() {

this.router.navigate(['/medicinelist']);

}

}

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { UpdateMedicineComponent } from './update-medicine.component';

describe('UpdateMedicineComponent', () => {

let component: UpdateMedicineComponent;

let fixture: ComponentFixture<UpdateMedicineComponent>;

beforeEach(async () => {

await TestBed.configureTestingModule({

declarations: [ UpdateMedicineComponent ]

})

.compileComponents();

fixture = TestBed.createComponent(UpdateMedicineComponent);

component = fixture.componentInstance;

fixture.detectChanges();

});

it('should create', () => {

expect(component).toBeTruthy();

});

});

<nav style="margin:auto" class="navbar navbar-expand-lg bg-dark">

<a style="color: white;" class="navbar-brand" href="#">APOLLO HOSPITALS</a>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav me-auto mb-2 mb-lg-0">

<li class="nav-item">

<a style="color: white;"class="nav-link active" aria-current="page" href="javascript:void(0);" routerLink="/home">Home</a>

</ul>

</div>

</nav>

<div class="container">

<hr>

<hr>

<hr>

<h3>Update Medicine</h3>

<form (ngSubmit) = "onSubmit()">

<div class="form-group">

<label> ID </label>

<input type="number" class="form-control" [(ngModel)]="medicine.id" id="id" name="id">

</div>

<div class="form-group">

<label> Drug Name</label>

<input type="text" class="form-control" [(ngModel)]="medicine.drugName" id="drugName" name="drugName">

</div>

<div class="form-group">

<label> Stock </label>

<input type="text" class="form-control" [(ngModel)]="medicine.stock" id="stock" name="stock">

</div>

<br>

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

</form>

</div>

.table

{

box-shadow: 10px -10px 5px #CCC;

}

.table

{

border-radius:25px

}

.table-header {

background-color: #95A5A6;

font-size: 14px;

text-transform: uppercase;

letter-spacing: 0.03em;

}

.table-header-row {

background-color: #95A5A6;

font-size: 14px;

text-transform: uppercase;

letter-spacing: 0.03em;

color: #fff;

font-weight: bold;

border-bottom: 1px solid #fff;

}

.navbar{

position: fixed;

top: 0;

width: 100%;

}

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

import { ActivatedRoute, Router } from '@angular/router';

import { Patient } from '../patient';

import { PatientService } from '../patient.service';

@Component({

selector: 'app-update-patient',

templateUrl: './update-patient.component.html',

styleUrls: ['./update-patient.component.css']

})

export class UpdatePatientComponent implements OnInit {

id: number;

patient: Patient = new Patient();

constructor(private patientService: PatientService,

private route: ActivatedRoute,

private router: Router) { }

ngOnInit(): void {

this.id = this.route.snapshot.params['id'];

this.patientService.getPatientById(this.id).subscribe(data => {

this.patient = data;

}, error => console.log(error) );

}

onSubmit() {

this.patientService.updatePatient(this.id, this.patient).subscribe(data => {

this.goToPatientlist();

}

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

}

goToPatientlist() {

this.router.navigate(['/docdash']);

}

}

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { UpdatePatientComponent } from './update-patient.component';

describe('UpdatePatientComponent', () => {

let component: UpdatePatientComponent;

let fixture: ComponentFixture<UpdatePatientComponent>;

beforeEach(async () => {

await TestBed.configureTestingModule({

declarations: [ UpdatePatientComponent ]

})

.compileComponents();

fixture = TestBed.createComponent(UpdatePatientComponent);

component = fixture.componentInstance;

fixture.detectChanges();

});

it('should create', () => {

expect(component).toBeTruthy();

});

});

<nav style="margin:auto" class="navbar navbar-expand-lg bg-dark">

<a style="color: white;" class="navbar-brand" href="#">APOLLO HOSPITALS</a>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav me-auto mb-2 mb-lg-0">

</ul>

</div>

</nav>

<div class="container">

<div class="col-md-6-offset-md-3">

<hr>

<hr>

<hr>

<h3>Update Patient</h3>

<form (ngSubmit) = "onSubmit()">

<div class="form-group">

<label>ID</label>

<input type="number" class="form-control" id="id" [(ngModel)]="patient.id" name="id" >

</div>

<div class="form-group">

<label>Name</label>

<input type="text" class="form-control" name="name" [(ngModel)]="patient.name" name="name" >

</div>

<div class="form-group">

<label>Age</label>

<input type="text" class="form-control" name="age" [(ngModel)]="patient.age" name="age" >

</div>

<div class="form-group">

<label>Blood Type</label>

<input type="text" class="form-control" name="blood" [(ngModel)]="patient.blood" name="blood" >

</div>

<div class="form-group">

<label>Prescription</label>

<input type="text" class="form-control" name="prescription" [(ngModel)]="patient.prescription" name="prescription" >

</div>

<div class="form-group">

<label>Dosage</label>

<input type="text" class="form-control" name="dose" [(ngModel)]="patient.dose" name="dose" >

</div>

<div class="form-group">

<label>Fees</label>

<input type="text" class="form-control" name="fees" [(ngModel)]="patient.fees" name="fees" >

</div>

<div class="form-group">

<label>Urgency</label>

<input type="text" class="form-control" name="urgency" [(ngModel)]="patient.urgency" name="urgency" >

</div>

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

</form>

</div>

</div>

.table

{

box-shadow: 10px -10px 5px #CCC;

}

.table

{

border-radius:25px

}

.table-header {

background-color: #95A5A6;

font-size: 14px;

text-transform: uppercase;

letter-spacing: 0.03em;

}

.table-header-row {

background-color: #95A5A6;

font-size: 14px;

text-transform: uppercase;

letter-spacing: 0.03em;

color: #fff;

font-weight: bold;

border-bottom: 1px solid #fff;

}

.navbar{

position: fixed;

top: 0;

width: 100%;

}

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

import { ActivatedRoute } from '@angular/router';

import { PatientService } from '../patient.service';

import { Patient } from '../patient';

@Component({

selector: 'app-view-patient',

templateUrl: './view-patient.component.html',

styleUrls: ['./view-patient.component.css']

})

export class ViewPatientComponent implements OnInit {

id: number;

patient: Patient

constructor(private route: ActivatedRoute, private patientService: PatientService) { }

ngOnInit(): void {

this.id = this.route.snapshot.params['id'];

this.patient = new Patient();

this.patientService.getPatientById(this.id).subscribe(data => {

this.patient = data;

} );

}

}

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { ViewPatientComponent } from './view-patient.component';

describe('ViewPatientComponent', () => {

let component: ViewPatientComponent;

let fixture: ComponentFixture<ViewPatientComponent>;

beforeEach(async () => {

await TestBed.configureTestingModule({

declarations: [ ViewPatientComponent ]

})

.compileComponents();

fixture = TestBed.createComponent(ViewPatientComponent);

component = fixture.componentInstance;

fixture.detectChanges();

});

it('should create', () => {

expect(component).toBeTruthy();

});

});

<nav style="margin:auto" class="navbar navbar-expand-lg bg-dark">

<a style="color: white;" class="navbar-brand" href="#">APOLLO HOSPITALS</a>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav me-auto mb-2 mb-lg-0">

<li class="nav-item">

<a style="color: white;"class="nav-link active" aria-current="page" href="javascript:void(0);" routerLink="/docdash">Back</a>

</ul>

</div>

</nav>

<hr>

<hr>

<hr>

<div class="container">

<h3>Patient Diagnosis Details</h3>

<div>

<div>

<label> <b> Name: </b></label> {{patient.name}}

</div>

<div>

<label> <b> Age: </b></label> {{patient.age}}

</div>

<div>

<label> <b> Blood Type: </b></label> {{patient.blood}}

</div>

<div>

<label> <b> Dosage </b></label> {{patient.dose}}

</div>

<div>

<label> <b> Urgency </b></label> {{patient.urgency}}

</div>

<div>

<label> <b> Detailed Diagnosis & Prescription: </b></label> {{patient.prescription}}

</div>

</div>

</div>

.table

{

box-shadow: 10px -10px 5px #CCC;

}

.table

{

border-radius:25px

}

.table-header {

background-color: #95A5A6;

font-size: 14px;

text-transform: uppercase;

letter-spacing: 0.03em;

}

.table-header-row {

background-color: #95A5A6;

font-size: 14px;

text-transform: uppercase;

letter-spacing: 0.03em;

color: #fff;

font-weight: bold;

border-bottom: 1px solid #fff;

}

.navbar{

position: fixed;

top: 0;

width: 100%;

}

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

@Component({

selector: 'app-newsfeed',

templateUrl: './newsfeed.component.html',

styleUrls: ['./newsfeed.component.css']

})

export class NewsfeedComponent implements OnInit {

constructor() { }

ngOnInit(): void {

}

}

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { NewsfeedComponent } from './newsfeed.component';

describe('NewsfeedComponent', () => {

let component: NewsfeedComponent;

let fixture: ComponentFixture<NewsfeedComponent>;

beforeEach(async () => {

await TestBed.configureTestingModule({

declarations: [ NewsfeedComponent ]

})

.compileComponents();

fixture = TestBed.createComponent(NewsfeedComponent);

component = fixture.componentInstance;

fixture.detectChanges();

});

it('should create', () => {

expect(component).toBeTruthy();

});

});

<!doctype html>

<html lang="en">

<head>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"

integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin="anonymous">

<title>Apollo Hospitals</title>

</head>

<body>

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

<a class="navbar-brand" href="#"> <b> APOLLO HOSPITALS </b></a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent"

aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav mr-auto">

</ul>

<div class="mx-2">

<button style="margin: 5px;" class="btn btn-danger" data-toggle="modal" data-target="#loginModal"><a style="color: white;" href="javascript:void(0);" routerLink="/doclogin">Login as Doctor</a></button>

<button style="margin: 5px;"class="btn btn-danger" data-toggle="modal" data-target="#loginModal"><a style="color:white;" href="javascript:void(0);" routerLink="/adlogin">Login as Admin</a></button>

</div>

</div>

</nav>

.carousel-caption {

color: rgb(255, 255, 255);

text-shadow: -1px 1px 0 #000,

1px 1px 0 #000,

1px -1px 0 #000,

-1px -1px 0 #000;

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

import { Router } from '@angular/router';

import { Patient } from '../patient';

import { PatientService } from '../patient.service';

@Component({

selector: 'app-docdash',

templateUrl: './docdash.component.html',

styleUrls: ['./docdash.component.css']

})

export class DocdashComponent implements OnInit {

searchText: string;

patients: Patient[];

constructor(private patientService: PatientService,

private router: Router) { }

ngOnInit(): void {

this.getPatients();

}

private getPatients(){

this.patientService.getPatientslist().subscribe(data => { this.patients = data;

});

}

viewPatient(id: number) {

this.router.navigate(['viewpatient', id]);

}

updatePatient(id: number) {

this.router.navigate(['updatepatient', id]);

}

deletePatient(id: number) {

this.patientService.deletePatient(id).subscribe(data => {

console.log(data);

this.getPatients();

} );

}

}

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { DocdashComponent } from './docdash.component';

describe('DocdashComponent', () => {

let component: DocdashComponent;

let fixture: ComponentFixture<DocdashComponent>;

beforeEach(async () => {

await TestBed.configureTestingModule({

declarations: [ DocdashComponent ]

})

.compileComponents();

fixture = TestBed.createComponent(DocdashComponent);

component = fixture.componentInstance;

fixture.detectChanges();

});

it('should create', () => {

expect(component).toBeTruthy();

});

});

<nav style="margin:auto" class="navbar navbar-expand-lg bg-dark">

<a style="color: white;" class="navbar-brand" href="#">APOLLO HOSPITALS</a>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav me-auto mb-2 mb-lg-0">

<li class="nav-item">

<a style="color: white;"class="nav-link active" aria-current="page" href="javascript:void(0);" routerLink="/createpatient">Add Patient</a>

</li>

<li class="nav-item">

<a style="color: white;"class="nav-link active" aria-current="page" href="javascript:void(0);" routerLink="/medicinelist">View Medicines</a>

</li>

</ul>

<div class="mx-2">

<button style="margin: 5px;"class="btn btn-danger" data-toggle="modal" data-target="#loginModal"><a style="color:white;" href="javascript:void(0);" routerLink="/home">Logout</a></button>

</div>

</div>

</nav>

<div class="grid">

<div class="container">

<hr>

<hr>

<hr>

<h2>Dashboard</h2>

<a style="color: rgb(86, 86, 86); font-size: x-large;"><marquee behavior="scroll" direction="left">

Welcome to Dashboard. Please find the patient list available below!!. Click Add Patient to Add patients, Click View Medicines to view and update Medicines</marquee></a>

<hr>

<input type="text" name="search" [(ngModel)]="searchText" placeholder="Search">

<br>

<a style="color: rgb(86, 86, 86); font-size: xx-large;">Current Patient List in your Queue</a>

<div class="tableFixHead">

<br>

<table class="table table-striped">

<thead class="table-header">

<tr>

<th>Id</th>

<th>Name</th>

<th>Age</th>

<th>Blood type</th>

<th>Urgency</th>

<th>Action</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let patient of patients | filter:searchText">

<td>{{patient.id}}</td>

<td>{{patient.name}}</td>

<td>{{patient.age}}</td>

<td>{{patient.blood}}</td>

<td>{{patient.urgency}}</td>

<td>

<button (click) = "updatePatient(patient.id)" class="btn btn-info">Update</button>

<button (click) = "deletePatient(patient.id)" class="btn btn-danger" style="margin-left: 10px;">Delete</button>

<button (click) = "viewPatient(patient.id)" class="btn btn-success" style="margin-left: 10px;">View</button>

</td>

</tr>

</tbody>

</table>

</div>

</div>

.table

{

box-shadow: 10px -10px 5px #CCC;

}

.table-header {

background-color: #95A5A6;

font-size: 14px;

text-transform: uppercase;

letter-spacing: 0.03em;

}

.table-header-row {

background-color: #95A5A6;

font-size: 14px;

text-transform: uppercase;

letter-spacing: 0.03em;

color: #fff;

font-weight: bold;

border-bottom: 1px solid #fff;

}

.navbar{

position: fixed;

top: 0;

width: 100%;

}

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

import { Router } from '@angular/router';

import { AuthenticationService } from '../authentication.service';

@Component({

selector: 'app-doclogin',

templateUrl: './doclogin.component.html',

styleUrls: ['./doclogin.component.css']

})

export class DocloginComponent implements OnInit {

username = 'user'

password = ''

invalidLogin = false

constructor(private router:Router, public loginservice: AuthenticationService) { }

ngOnInit(): void {

}

checkLogin() {

if (this.loginservice.authenticate(this.username, this.password)

) {

this.router.navigate(['docdash'])

this.invalidLogin = false

} else

{

this.invalidLogin = true

alert("Wrong Credentials");

this.router.navigate(['home'])

}

}

}

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { DocloginComponent } from './doclogin.component';

describe('DocloginComponent', () => {

let component: DocloginComponent;

let fixture: ComponentFixture<DocloginComponent>;

beforeEach(async () => {

await TestBed.configureTestingModule({

declarations: [ DocloginComponent ]

})

.compileComponents();

fixture = TestBed.createComponent(DocloginComponent);

component = fixture.componentInstance;

fixture.detectChanges();

});

it('should create', () => {

expect(component).toBeTruthy();

});

});

<nav style="margin:auto" class="navbar navbar-expand-lg bg-dark">

<a style="color: white;" class="navbar-brand" href="#">APOLLO HOSPITALS</a>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav me-auto mb-2 mb-lg-0">

<li class="nav-item">

<a style="color: white;"class="nav-link active" aria-current="page" href="javascript:void(0);" routerLink="/home">Home</a>

</li>

</ul>

<form class="d-flex" role="search">

<input class="form-control me-2" type="search" placeholder="Search" aria-label="Search">

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

</form>

</div>

</nav>

<div class="container">

<div class="row">

<div class="col-md-6">

<div class="card">

<div class="text-center">

<h1>Login As Doctor</h1>

<h6>Please enter your Username & Password</h6>

</div>

<form>

<div class="mb-3">

<label for="formGroupExampleInput" class="form-label"><h4>Username</h4></label>

<input type="text" name="username" [(ngModel)]="username" class="form-control" id="formGroupExampleInput" placeholder="User name" required>

</div>

<div class="mb-3">

<label for="formGroupExampleInput2" class="form-label"><h4>Password</h4></label>

<input type="password" name="password" [(ngModel)]="password" class="form-control" id="formGroupExampleInput2" placeholder="Password" required>

</div>

<div class="d-grid gap-2 d-md-block">

<button (click)=checkLogin() style="margin: 5px;" class="btn btn-success" type="button" >Login</button>

<button style="margin: 5px;" class="btn btn-warning" type="button" routerLink="/home">Cancel</button>

</div>

</form>

</div>

</div>

</div>

</div>

.card{

border: none;

border-radius: 30px;

width: 500px;

padding: 40px;

top: 40%;

left: 30%;

transform: translate(30%, -25%);

background: rgb(158, 219, 219);

justify-content: center;

color: navy;

justify-items: center;

box-shadow: 10px 10px 10px rgb(120, 120, 120);

}