

<p>MEDICARE CAPSTONE PROJECT BY ROHAN KARANDE</p>
--

DATE OF SUBMISSION – 23/12/2022

GITHUB LINK - https://github.com/codewithrohan10/PHASE5_CAPSTONE_MEDICARE.git

BACKEND SOURCE CODE -

MedicareAppApplication.java

```
-----
package com.simplilearn.Medicare_App;

import java.util.HashSet;
import java.util.Set;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import com.simplilearn.Medicare_App.exceptions.UserFoundException;
import com.simplilearn.Medicare_App.model.Role;
import com.simplilearn.Medicare_App.model.User;
import com.simplilearn.Medicare_App.model.UserRole;
import com.simplilearn.Medicare_App.service.UserService;

@SpringBootApplication
public class MedicareAppApplication implements CommandLineRunner{

    @Autowired
    private UserService userService;
    @Autowired
    private BCryptPasswordEncoder bCryptPasswordEncoder;

    public static void main(String[] args) {
        SpringApplication.run(MedicareAppApplication.class, args);
    }
    @Override
    public void run(String... args) throws Exception {

        System.out.println("code started");
    }
}
```

JwtAuthenticationEntryPoint.java

```
package com.simplilearn.Medicare_App.config;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.security.core.AuthenticationException;
import org.springframework.security.web.AuthenticationEntryPoint;
import org.springframework.stereotype.Component;

@Component
public class JwtAuthenticationEntryPoint implements AuthenticationEntryPoint {

    @Override
    public void commence(HttpServletRequest request, HttpServletResponse response,
        AuthenticationException authException) throws IOException, ServletException {

        response.sendError(HttpServletResponse.SC_UNAUTHORIZED, "Unauthorized : Server");
    }

}
```

JwtAuthenticationFilter.java

```
package com.simplilearn.Medicare_App.config;

import java.io.IOException;

import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;
import org.springframework.stereotype.Component;
```

```

import org.springframework.web.filter.OncePerRequestFilter;

import com.simplilearn.Medicare_App.service.impl.UserDetailsServiceImp;

import io.jsonwebtoken.ExpiredJwtException;

@Component
public class JwtAuthenticationFilter extends OncePerRequestFilter{

@Autowired
private UserDetailsServiceImp userDetailsServiceImp;

@Autowired
private JwtUtils jwtUtils;

@Override
protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response,
FilterChain filterChain)
throws ServletException, IOException {

final String requestTokenHeader = request.getHeader("Authorization");
System.out.printf("requestTokenHeader: ", requestTokenHeader);
String username = null;
String jwtToken = null;

if(requestTokenHeader != null && requestTokenHeader.startsWith("Bearer ")) {
//yes

jwtToken = requestTokenHeader.substring(7);
try {
username = this.jwtUtils.extractUsername(jwtToken);
} catch(ExpiredJwtException e) {
e.printStackTrace();
System.out.println("jwt token has expired");
}catch(Exception e) {
e.printStackTrace();
System.out.println("error");
}
}else {
System.out.println("Invlaid token, not start with bearer string");
}

//validated

if(username != null && SecurityContextHolder.getContext().getAuthentication()==null) {
final UserDetails userDetails = this.userDetailsServiceImp.loadUserByUsername(username);

if(this.jwtUtils.validateToken(jwtToken, userDetails)) {

```

```
//token is valid
```

```
UsernamePasswordAuthenticationToken usernamePasswordAuthenticationToken= new
UsernamePasswordAuthenticationToken(userDetails, null, userDetails.getAuthorities());
usernamePasswordAuthenticationToken.setDetails(new
WebAuthenticationDetailsSource().buildDetails(request));
SecurityContextHolder.getContext().setAuthentication(usernamePasswordAuthenticationToken);
}
}else {
System.out.println("Token is not valid");
}
filterChain.doFilter(request, response);

}

}
```

JwtUtils.java

```
-----
package com.simplilearn.Medicare_App.config;

import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.stereotype.Service;

import java.util.Date;
import java.util.HashMap;
import java.util.Map;
import java.util.function.Function;

@Service
public class JwtUtils {

    private String SECRET_KEY = "secret";

    public String extractUsername(String token) {
        return extractClaim(token, Claims::getSubject);
    }

    public Date extractExpiration(String token) {
        return extractClaim(token, Claims::getExpiration);
    }

    public <T> T extractClaim(String token, Function<Claims, T> claimsResolver) {
```

```

        final Claims claims = extractAllClaims(token);
        return claimsResolver.apply(claims);
    }
    private Claims extractAllClaims(String token) {
        return Jwts.parser().setSigningKey(SECRET_KEY).parseClaimsJws(token).getBody();
    }

    private Boolean isTokenExpired(String token) {
        return extractExpiration(token).before(new Date());
    }

    public String generateToken(UserDetails userDetails) {
        Map<String, Object> claims = new HashMap<>();
        return createToken(claims, userDetails.getUsername());
    }

    private String createToken(Map<String, Object> claims, String subject) {

        return Jwts.builder().setClaims(claims).setSubject(subject).setIssuedAt(new
Date(System.currentTimeMillis()))
            .setExpiration(new Date(System.currentTimeMillis() + 1000 * 60 * 60 * 10))
            .signWith(SignatureAlgorithm.HS256, SECRET_KEY).compact();
    }

    public Boolean validateToken(String token, UserDetails userDetails) {
        final String username = extractUsername(token);
        return (username.equals(userDetails.getUsername()) && !isTokenExpired(token));
    }
}

```

MySecurityConfig.java

```

package com.simplilearn.Medicare_App.config;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpMethod;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import
org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBu
ilder;
import
org.springframework.security.config.annotation.method.configuration.EnableGlobalMethodSecurit
y;

```

```

import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.NoOpPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

import com.simplilearn.Medicare_App.service.impl.UserDetailsServiceImpl;

@EnableWebSecurity
@Configuration
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class MySecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired
    private JwtAuthenticationEntryPoint unauthorizedHandler;

    @Autowired
    private UserDetailsServiceImpl userDetailsServiceImpl;

    @Autowired
    private JwtAuthenticationFilter jwtAuthenticationFilter;

    @Override
    @Bean
    public AuthenticationManager authenticationManagerBean() throws Exception {
        return super.authenticationManagerBean();
    }

    @Bean
    public BCryptPasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }

    //      @Bean
    //      public PasswordEncoder passwordEncoder() {
    //          return NoOpPasswordEncoder.getInstance();
    //      }

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {

        auth.userDetailsService(this.userDetailsServiceImpl).passwordEncoder(passwordEncoder());
    }

```

```

@Override
protected void configure(HttpSecurity http) throws Exception {

    http
    .csrf()
    .disable()
    .cors()
    .disable()
    .authorizeRequests()
    .antMatchers("/generate-token","/user/","/user/test").permitAll()
    .antMatchers(HttpMethod.OPTIONS).permitAll()
    .anyRequest().authenticated()
    .and()
    .exceptionHandling().authenticationEntryPoint(unauthorizedHandler)
    .and()
    .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS);

    http.addFilterBefore(jwtAuthenticationFilter, UsernamePasswordAuthenticationFilter.class);
}
}

```

AuthenticateController

```

package com.simplilearn.Medicare_App.controller;

import java.security.Principal;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.BadCredentialsException;
import org.springframework.security.authentication.DisabledException;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import com.simplilearn.Medicare_App.config.JwtUtils;
import com.simplilearn.Medicare_App.model.JwtRequest;
import com.simplilearn.Medicare_App.model.JwtResponse;
import com.simplilearn.Medicare_App.model.User;

```



```
import com.simplilearn.Medicare_App.service.impl.UserDetailsServiceImp;

@RestController
@CrossOrigin("http://localhost:4200")
public class AuthenticateController {

    @Autowired
    private AuthenticationManager authenticationManager;

    @Autowired
    private UserDetailsServiceImp userDetailsServiceImp;

    @Autowired
    private JwtUtils jwtUtils;

    //generate token

    @PostMapping("/generate-token")
    public ResponseEntity<?> generateToken(@RequestBody JwtRequest jwtRequest) throws
    Exception{

    try {

    authenticate(jwtRequest.getUsername(), jwtRequest.getPassword());

    }catch(UsernameNotFoundException e) {
    e.printStackTrace();
    throw new Exception("User not found");
    }

    UserDetails userDetails =
    this.userDetailsServiceImp.loadUserByUsername(jwtRequest.getUsername());
    String token = this.jwtUtils.generateToken(userDetails);

    return ResponseEntity.ok(new JwtResponse(token));

    }

    private void authenticate(String username, String password) throws Exception {

    try {
    authenticationManager.authenticate(new UsernamePasswordAuthenticationToken(username,
    password));
```

```

}catch(DisabledException e) {
throw new Exception("USER DISABLED "+e.getMessage());
}catch(BadCredentialsException e) {
throw new Exception("Invlaid Credentials " +e.getMessage());
}
}

//return details of current user
@GetMapping("/current-user")
public User getCurrentUser(Principal pricipal) {
return ((User)this.userDetailsServiceImpl.loadUserByUsername(pricipal.getName()));
}

}

```

CategoryController.java

```

package com.simplilearn.Medicare_App.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.simplilearn.Medicare_App.model.Category;
import com.simplilearn.Medicare_App.service.CategoryService;

@RestController
@CrossOrigin("http://localhost:4200")
@RequestMapping(path = "/category")
public class CategoryController {

    @Autowired
    private CategoryService categoryService;

    //add category
    @PostMapping("/add")

```

```

public ResponseEntity<?> addCategory(@RequestBody Category category){
    Category category1 = this.categoryService.addCategory(category);
    return ResponseEntity.ok(category1);
}

//get category
@GetMapping("/{categoryId}")
public Category getCategory(@PathVariable("categoryId") Long categoryId){
    return this.categoryService.getCategory(categoryId);
}

// get all categories

@GetMapping("/")
public ResponseEntity<?> getAllCategories(){
    return ResponseEntity.ok(this.categoryService.getCategories());
}

@PutMapping("/")
public Category updateCategory(@RequestBody Category category) {
    return this.categoryService.updateCategory(category);
}

@DeleteMapping("/{categoryId}")
public void deleteCategory(@PathVariable("categoryId") Long categoryId) {
    this.categoryService.deleteCategory(categoryId);
}

}

```

MedicineController.java

```

package com.simplilearn.Medicare_App.controller;

import java.io.IOException;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;

```

```

import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.multipart.MultipartFile;

import com.simplilearn.Medicare_App.model.Category;
import com.simplilearn.Medicare_App.model.Medicine;
import com.simplilearn.Medicare_App.repository.MedicineRepository;
import com.simplilearn.Medicare_App.service.MedicineService;

@RestController
@CrossOrigin("http://localhost:4200")
@RequestMapping(path = "medicines")
public class MedicineController {

    @Autowired
    private MedicineService medicineService;
    //private byte[] bytes;

    //add category
    @PostMapping("/add")
    public ResponseEntity<?> addMedicine(@RequestBody Medicine medicine){
        Medicine medicine1 = this.medicineService.addMedicine(medicine);
        return ResponseEntity.ok(medicine1);
    }

    //get category
    @GetMapping("/{medicineId}")
    public Medicine getMedicine(@PathVariable("medicineId") Long medicineId){
        return this.medicineService.getMedicine(medicineId);
    }

    // get all categories

    @GetMapping("/")
    public ResponseEntity<?> getAllMedicines(){
        return ResponseEntity.ok(this.medicineService.getMedicines());
    }

    @PutMapping("/")
    public Medicine updateMedicine(@RequestBody Medicine medicine) {
        return this.medicineService.updateMedicine(medicine);
    }
}

```

```

@DeleteMapping("/{id}")
public Medicine deleteMedicine(@PathVariable("id") Long id) {
    return this.medicineService.deleteMedicine(id);
}

//      @DeleteMapping("/{name}")
//      public void deleteMedicine(@PathVariable("name") String name) {
//          this.medicineService.deleteByName(name);
//      }

@GetMapping("/category/{cid}")
public List<Medicine> getMedicinesOfCategory(@PathVariable("cid") Long cid){

    Category category = new Category();
    category.setCid(cid);
    return this.medicineService.MedicinesOfCategory(category);

}

@GetMapping("/active")
public List<Medicine> getActiveMedicines(){
    return this.medicineService.getActiveMedicine();
}

@GetMapping("/category/active/{cid}")
public List<Medicine> getActiveMedicinesOfCategory(@PathVariable("cid") Long cid){

    Category category = new Category();
    category.setCid(cid);
    return this.medicineService.getActiveMedicinesOfCategory(category);

}

@GetMapping("/active/{name}")
public List<Medicine> searchByName(@PathVariable("name") String name){
    return this.medicineService.searchByName(name);
}
}

```

UserController.java

package com.simplilearn.Medicare_App.controller;

```

import java.util.HashSet;
import java.util.Optional;
import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.simplilearn.Medicare_App.exceptions.UserNotFoundException;
import com.simplilearn.Medicare_App.exceptions.UserNotFoundException;
import com.simplilearn.Medicare_App.model.Role;
import com.simplilearn.Medicare_App.model.User;
import com.simplilearn.Medicare_App.model.UserRole;
import com.simplilearn.Medicare_App.service.UserService;

@RestController
@CrossOrigin("http://localhost:4200")
@RequestMapping(path = "/user")
public class UserController {

    @Autowired
    private UserService userService;

    @Autowired
    private BCryptPasswordEncoder bCryptPasswordEncoder;

    @GetMapping("/test")
    public String test() {
        return "Welcome to Medicare";
    }

    @PostMapping("/")
    public User createUser(@RequestBody User user) throws Exception {

        //encoding password with bcryptpasswordencoder
        user.setPassword(this.bCryptPasswordEncoder.encode(user.getPassword()));
    }

```

```

Set<UserRole> roles = new HashSet<UserRole>();
Role role = new Role(25L, "NORMAL");
UserRole userRole = new UserRole();
userRole.setUser(user);
userRole.setRole(role);
roles.add(userRole);
user.setProfile("default.png");
return this.userService.createUser(user, roles);
}

@GetMapping("/{username}")
public User getUser(@PathVariable(value = "username") String username) {
return this.userService.getUser(username);
}

@DeleteMapping("{userId}")
public void deleteUser(@PathVariable("userId") Long userId) {
this.userService.deleteUser(userId);
}

@ExceptionHandler(UserFoundException.class)
public ResponseEntity<?> exceptionHandler(UserFoundException ex){
return ResponseEntity.ok(ex.getMessage());
}
}

```

UserFoundException.java

```

-----
package com.simplilearn.Medicare_App.exceptions;

public class UserFoundException extends Exception {

/**
 *
 */
private static final long serialVersionUID = 1L;

public UserFoundException() {
super("User with this username is already there in database!! Try with another username");
}

public static long getSerialVersionUID() {
return serialVersionUID;
}
}

```

```

}

public UserFoundException(String message, Throwable cause) {
    super(message, cause);
    // TODO Auto-generated constructor stub
}

public UserFoundException(String message) {
    super(message);
    // TODO Auto-generated constructor stub
}

public UserFoundException(Throwable cause) {
    super(cause);
    // TODO Auto-generated constructor stub
}

}

```

UserNotFoundException.java

```

-----
package com.simplilearn.Medicare_App.exceptions;

public class UserNotFoundException extends Exception {

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    public UserNotFoundException() {
        super("User with this username not found in database");
    }

    public UserNotFoundException(String message, Throwable cause, boolean enableSuppression,
        boolean writableStackTrace) {
        super(message, cause, enableSuppression, writableStackTrace);
        // TODO Auto-generated constructor stub
    }

    public UserNotFoundException(String message, Throwable cause) {
        super(message, cause);
        // TODO Auto-generated constructor stub
    }
}

```



```
public UserNotFoundException(String message) {
    super(message);
    // TODO Auto-generated constructor stub
}

public UserNotFoundException(Throwable cause) {
    super(cause);
    // TODO Auto-generated constructor stub
}

}
```

Authority.java

```
-----
package com.simplilearn.Medicare_App.model;

import org.springframework.security.core.GrantedAuthority;

public class Authority implements GrantedAuthority{

    private static final long serialVersionUID = 1L;
    private String authority;

    public Authority(String authority) {
        super();
        this.authority = authority;
    }

    public Authority() {
        super();
        // TODO Auto-generated constructor stub
    }

    @Override
    public String getAuthority() {
        // TODO Auto-generated method stub
        return this.authority;
    }

}
```

Category.java

```
package com.simplilearn.Medicare_App.model;

import java.util.LinkedHashSet;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.Table;

import com.fasterxml.jackson.annotation.JsonIgnore;

@Entity
@Table(name = "category")
public class Category {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long cid;
    private String title;
    private String description;

    @OneToMany(mappedBy = "category", cascade = CascadeType.ALL)
    @JsonIgnore
    private Set<Medicine> medicines = new LinkedHashSet<>();

    public Set<Medicine> getMedicines() {
        return medicines;
    }
    public void setMedicines(Set<Medicine> medicines) {
        this.medicines = medicines;
    }
    public Category() {
        super();
    }
    public Category(String title, String description) {
        super();
        this.title = title;
        this.description = description;
    }
    public Long getCid() {
        return cid;
    }
    public void setCid(Long cid) {
```

```

this.cid = cid;
}
public String getTitle() {
return title;
}
public void setTitle(String title) {
this.title = title;
}
public String getDescription() {
return description;
}
public void setDescription(String description) {
this.description = description;
}
}

```

JwtRequest.java

```

-----
package com.simplilearn.Medicare_App.model;

public class JwtRequest {

    String username;
    String password;
    public JwtRequest() {
    }
    public JwtRequest(String username, String password) {
    super();
    this.username = username;
    this.password = password;
    }
    public String getUsername() {
    return username;
    }
    public void setUsername(String username) {
    this.username = username;
    }
    public String getPassword() {
    return password;
    }
    public void setPassword(String password) {
    this.password = password;
    }
}

```

```
}
```

JwtResponse.java

```
package com.simplilearn.Medicare_App.model;
```

```
public class JwtResponse {  
    String token;
```

```
    public JwtResponse(String token) {  
        this.token = token;  
    }
```

```
    public JwtResponse() {  
  
    }
```

```
    public String getToken() {  
        return token;  
    }
```

```
    public void setToken(String token) {  
        this.token = token;  
    }
```

```
}
```

Medicine.java

```
package com.simplilearn.Medicare_App.model;
```

```
import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.FetchType;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;  
import javax.persistence.ManyToOne;  
import javax.persistence.Table;
```

```
@Entity
```

```
public class Medicine {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;

    public boolean isActive() {
        return active;
    }

    public void setActive(boolean active) {
        this.active = active;
    }

    private String name;
    private String seller;
    @Column(length= 5000)
    private String prodDesc;
    private String offers;
    private long price;
    private boolean active = false;
    //      @Column(name = "picByte", length= 1000)
    //      private byte[] picByte;

    @ManyToOne(fetch = FetchType.EAGER)
    private Category category;

    public Medicine() {
    }

    public Medicine(long id, String name, String seller, String prodDesc, String offers, long price,
        boolean active,
        Category category) {
        super();
        this.id = id;
        this.name = name;
        this.seller = seller;
        this.prodDesc = prodDesc;
        this.offers = offers;
        this.price = price;
        this.active = active;
        this.category = category;
    }
}
```

```
//      public byte[] getPicByte() {  
//          return picByte;  
//      }  
//  
//  
//      public void setPicByte(byte[] picByte) {  
//          this.picByte = picByte;  
//      }
```

```
public long getId() {  
    return id;  
}
```

```
public void setId(long id) {  
    this.id = id;  
}
```

```
public Category getCategory() {  
    return category;  
}
```

```
public void setCategory(Category category) {  
    this.category = category;  
}
```

```
public String getName() {  
    return name;  
}
```

```
public void setName(String name) {  
    this.name = name;  
}
```

```
public String getSeller() {  
    return seller;  
}
```

```
public void setSeller(String seller) {  
    this.seller = seller;  
}
```

```
public String getProdDesc() {  
    return prodDesc;  
}
```

```

public void setProdDesc(String prodDesc) {
this.prodDesc = prodDesc;
}

public String getOffers() {
return offers;
}

public void setOffers(String offers) {
this.offers = offers;
}

public long getPrice() {
return price;
}

public void setPrice(long price) {
this.price = price;
}

}

```

Role.java

```

package com.simplilearn.Medicare_App.model;

import java.util.HashSet;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.Id;
import javax.persistence.OneToOne;
import javax.persistence.Table;

import com.fasterxml.jackson.annotation.JsonIgnore;

@Entity
@Table(name ="roles")
public class Role {

```

```

@Id
private Long roleId;
private String roleName;

@OneToMany(cascade = CascadeType.ALL, fetch = FetchType.LAZY, mappedBy = "role")
@JsonIgnore
private Set<UserRole> userRoles = new HashSet<>();
public Role() {
}
public Role(Long roleId, String roleName) {
super();
this.roleId = roleId;
this.roleName = roleName;
}

public Set<UserRole> getUserRoles() {
return userRoles;
}
public void setUserRoles(Set<UserRole> userRoles) {
this.userRoles = userRoles;
}
public Long getRoleId() {
return roleId;
}
public void setRoleId(Long roleId) {
this.roleId = roleId;
}
public String getRoleName() {
return roleName;
}
public void setRoleName(String roleName) {
this.roleName = roleName;
}
}

```

User.java

```

-----
package com.simplilearn.Medicare_App.model;

import java.util.Collection;
import java.util.HashSet;
import java.util.Set;

```



```

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.Table;

import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

import com.fasterxml.jackson.annotation.JsonIgnore;

@Entity
@Table(name = "users")
public class User implements UserDetails {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String username;
    private String password;
    private String firstName;
    private String lastName;
    private String email;
    private String phone;
    private boolean enabled = true;
    private String profile;

    //one to many
    @OneToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER, mappedBy = "user")
    @JsonIgnore
    private Set<UserRole> userRoles = new HashSet<>();

    public User() {
    }

    public User(String username, String password, String firstName, String lastName, String email,
String phone,
String profile) {
super();

```

```
this.username = username;  
this.password = password;  
this.firstName = firstName;  
this.lastName = lastName;  
this.email = email;  
this.phone = phone;  
this.profile = profile;  
}
```

```
public Set<UserRole> getUserRoles() {  
    return userRoles;  
}
```

```
public void setUserRoles(Set<UserRole> userRoles) {  
    this.userRoles = userRoles;  
}
```

```
public String getProfile() {  
    return profile;  
}
```

```
public void setProfile(String profile) {  
    this.profile = profile;  
}
```

```
public User(Long id, String username, String password, String firstName, String lastName, String  
email,  
String phone, boolean enabled, String profile) {  
    super();  
    this.id = id;  
    this.username = username;  
    this.password = password;  
    this.firstName = firstName;  
    this.lastName = lastName;  
    this.email = email;  
    this.phone = phone;
```

```
this.enabled = enabled;  
this.profile = profile;  
}
```

```
public Long getId() {  
    return id;  
}
```

```
public void setId(Long id) {  
    this.id = id;  
}
```

```
public String getUsername() {  
    return username;  
}
```

```
public void setUsername(String username) {  
    this.username = username;  
}
```

```
public String getPassword() {  
    return password;  
}
```

```
public void setPassword(String password) {  
    this.password = password;  
}
```

```
public String getFirstName() {  
    return firstName;  
}
```

```
public void setFirstName(String firstName) {  
    this.firstName = firstName;  
}
```

```
public String getLastName() {  
    return lastName;  
}
```

```
public void setLastName(String lastName) {  
    this.lastName = lastName;  
}
```

```
public String getEmail() {  
    return email;  
}
```

```

}

public void setEmail(String email) {
    this.email = email;
}

public String getPhone() {
    return phone;
}

public void setPhone(String phone) {
    this.phone = phone;
}

public boolean isEnabled() {
    return enabled;
}

public void setEnabled(boolean enabled) {
    this.enabled = enabled;
}

@Override
public Collection<? extends GrantedAuthority> getAuthorities() {

    Set<Authority> set = new HashSet<>();

    this.userRoles.forEach(userRole -> {
        set.add(new Authority(userRole.getRole().getRoleName()));
    });

    return set;
}

@Override
public boolean isAccountNonExpired() {
    // TODO Auto-generated method stub
    return true;
}

@Override
public boolean isAccountNonLocked() {

```

```
// TODO Auto-generated method stub  
return true;  
}
```

```
@Override  
public boolean isCredentialsNonExpired() {  
// TODO Auto-generated method stub  
return true;  
}
```

```
}
```

UserRole.java

```
-----  
package com.simplilearn.Medicare_App.model;  
  
import javax.persistence.Entity;  
import javax.persistence.FetchType;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;  
import javax.persistence.ManyToOne;  
import javax.persistence.Table;  
  
@Entity  
@Table(name = "user_role")  
public class UserRole {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.AUTO)  
    private Long userRoleId;  
  
    @ManyToOne(fetch = FetchType.EAGER)  
    private User user;  
  
    @ManyToOne  
    private Role role;  
  
    public UserRole() {  
        super();  
        // TODO Auto-generated constructor stub  
    }
```

```
public Long getUserRoleId() {
return userRoleId;
}

public void setUserRoleId(Long userRoleId) {
this.userRoleId = userRoleId;
}

public User getUser() {
return user;
}

public void setUser(User user) {
this.user = user;
}

public Role getRole() {
return role;
}

public void setRole(Role role) {
this.role = role;
}

}


```

application.properties

```
spring.datasource.url=jdbc:mysql://localhost:3306/medicare_db?serverTimezone=UTC
spring.datasource.username=root
spring.datasource.password=Rohan10$
spring.jpa.hibernate.ddl-auto=update
spring.jackson.serialization.fail-on-empty-beans=false
spring.jpa.properties.hibernate.dialect= org.hibernate.dialect.MySQL8Dialect
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
```

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
```

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<parent>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-parent</artifactId>
<version>2.5.5</version>
<relativePath /> <!-- lookup parent from repository -->
</parent>
<groupId>com.simplilearn</groupId>
<artifactId>Medicare_App</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>Medicare_App</name>
<description>Demo project for Spring Boot</description>
<properties>
<java.version>17</java.version>
</properties>
<dependencies>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>
<!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-security -->
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-security</artifactId>
</dependency>
<!-- https://mvnrepository.com/artifact/io.jsonwebtoken/jjwt -->
<dependency>
<groupId>io.jsonwebtoken</groupId>
<artifactId>jjwt</artifactId>
<version>0.9.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/javax.xml.bind/jaxb-api -->
<dependency>
<groupId>javax.xml.bind</groupId>
<artifactId>jaxb-api</artifactId>
</dependency>

<dependency>
<groupId>mysql</groupId>
```

```

<artifactId>mysql-connector-java</artifactId>
<scope>runtime</scope>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-test</artifactId>
<scope>test</scope>
</dependency>

<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-chrome-driver -->
<dependency>
<groupId>org.seleniumhq.selenium</groupId>
<artifactId>selenium-chrome-driver</artifactId>
<version>3.141.59</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-firefox-driver -->
<dependency>
<groupId>org.seleniumhq.selenium</groupId>
<artifactId>selenium-firefox-driver</artifactId>
<version>3.141.59</version>
</dependency>

<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
<!-- <version>8.0.26</version> -->
</dependency>

</dependencies>

<build>
<plugins>
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
</plugin>
</plugins>
</build>

</project>

```

FRONTEND SOURCE CODE-

app-routing.module.ts


```

-----

import { PaymentComponent } from './pages/user/shopping-cart/payment/payment.component';
import { ShoppingCartComponent } from './pages/user/shopping-cart/shopping-cart.component';
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { AddCategoryComponent } from './pages/admin/add-category/add-category.component';
import { DashboardComponent } from './pages/admin/dashboard/dashboard.component';
import { ViewCategoriesComponent } from './pages/admin/view-categories/view-
categories.component';
import { WelcomeComponent } from './pages/admin/welcome/welcome.component';
import { HomeComponent } from './pages/home/home.component';
import { LoginComponent } from './pages/login/login.component';
import { ProfileComponent } from './pages/profile/profile.component';
import { SignupComponent } from './pages/signup/signup.component';
import { UserDashboardComponent } from './pages/user/user-dashboard/user-
dashboard.component';
import { ViewMedicinesComponent } from './pages/admin/view-medicines/view-
medicines.component';
import { AdminGuard } from './service/admin.guard';
import { NormalGuard } from './service/normal.guard';
import { AddMedicineComponent } from './pages/admin/add-medicine/add-medicine.component';
import { UpdateMedicinesComponent } from './pages/admin/update-medicines/update-
medicines.component';
import { UserWelcomeComponent } from './pages/user/user-welcome/user-welcome.component';

const routes: Routes = [
  { path: '', component: HomeComponent, pathMatch: 'full' },
  { path: 'signup', component: SignupComponent, pathMatch: 'full' },
  { path: 'login', component: LoginComponent, pathMatch: 'full' },
  { path: 'admin', component: DashboardComponent, canActivate:[AdminGuard],
    children:[{ path : '', component: WelcomeComponent},
      { path : 'profile', component: ProfileComponent},
      { path : 'categories', component: ViewCategoriesComponent},
      { path : 'add-category', component: AddCategoryComponent},
      { path : 'medicines', component: ViewMedicinesComponent},
      { path : 'add-medicine', component: AddMedicineComponent},
      { path : 'medicines/:id', component: UpdateMedicinesComponent},

    ]},
  { path: 'user-dashboard', component: UserDashboardComponent, canActivate:[NormalGuard],
    children:[{ path : '', component:UserWelcomeComponent},
      { path : 'profile', component:ProfileComponent},
      { path : 'payment', component:PaymentComponent},
      { path : 'search/:keyword', component:ShoppingCartComponent},
      { path : ':catId', component:ShoppingCartComponent}
    ]
  ]

```

```

}

];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }

```

admin.guard.ts

```

import { Injectable } from '@angular/core';
import { ActivatedRouteSnapshot, CanActivate, Router, RouterStateSnapshot, UrlTree } from
 '@angular/router';
import { Observable } from 'rxjs';
import { LoginService } from './login.service';

@Injectable({
  providedIn: 'root'
})
export class AdminGuard implements CanActivate {

  constructor(private login: LoginService, private router : Router){}

  canActivate(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> |
    boolean | UrlTree {

    if(this.login.isLoggedIn() && this.login.getUserRole() == 'ADMIN'){
      return true;
    }
    this.router.navigate(['login']);
    return false;
  }
}

```

auth.interceptor.ts

```
import { HttpEvent, HttpHandler, HttpInterceptor, HttpRequest, HTTP_INTERCEPTORS } from
"@angular/common/http";
import { Injectable } from "@angular/core";
import { Observable } from "rxjs";
import { LoginService } from "../login.service";
```

```
@Injectable()
```

```
export class AuthInterceptor implements HttpInterceptor{
```

```
  constructor(private login : LoginService){}
```

```
  intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
```

```
    let authReq = req;
    const token = this.login.getToken();
    if(token!=null){
      authReq = authReq.clone({
        setHeaders: {Authorization: `Bearer ${token}` },
      });
    }
    return next.handle(authReq);
  }
}
```

```
export const authInterceptorProviders = [
  {
    provide: HTTP_INTERCEPTORS,
    useClass: AuthInterceptor,
    multi: true
  },
];
```

```
normal.guard.ts
```

```
import { Injectable } from '@angular/core';
import { ActivatedRouteSnapshot, CanActivate, Router, RouterStateSnapshot, UrlTree } from
'@angular/router';
import { Observable } from 'rxjs';
import { LoginService } from '../login.service';
```

```
@Injectable({
  providedIn: 'root'
})
```

```
export class NormalGuard implements CanActivate {
```

```

constructor(private login: LoginService, private router : Router){}

canActivate(
  route: ActivatedRouteSnapshot,
  state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> |
boolean | UrlTree {
  if(this.login.isLoggedIn() && this.login.getUserRole() == 'NORMAL'){
    return true;
  }
  this.router.navigate(['login']);
  return false; }
}

```

login.component.html

```

<div class="loginPage" class = "bootstrap-wrapper">
  <div class="container">
    <div class="row mt20" style="margin-left: 330px">
      <div class="col-md-6 offset-mid-3">
        <mat-card>
          <div class="container text-center">
            
          </div>
          <h1 class="text-center">Login Here!!</h1>

          <!-- {{loginData | json}} -->
          <form (ngSubmit)="formSubmit()" >

            <mat-form-field class = "full-width "appearance="outline">
              <mat-label>Username</mat-label>
              <input
                [(ngModel)] = "loginData.username"
                name = "username"
                required
                matInput placeholder="Enter name"
              />
            </mat-form-field>

            <mat-form-field class = "full-width "appearance="outline">
              <mat-label>Password</mat-label>
              <input
                [(ngModel)] = "loginData.password"
                name = "password"
                required
                type = "password" matInput placeholder="Enter password">

```

```

        </mat-form-field>

        <div class="container text-center">
            <button mat-raised-button type = "submit" color="primary">Login</button>
            <button mat-raised-button class ="ml20" type = "submit"
color="accent">Reset</button>
        </div>
    </form>
</mat-card>
</div>
</div>
</div>
</div>

```

login.component.ts

```

import { Component, OnInit } from '@angular/core';
import { MatSnackBar } from '@angular/material/snack-bar';
import { Router } from '@angular/router';
import { LoginService } from 'src/app/service/login.service';
import Swal from 'sweetalert2';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {

  loginData = {
    username:"",
    password:"
  };

  constructor( private _snack : MatSnackBar, private login:LoginService, private router: Router) { }

  ngOnInit(): void {
  }

  formSubmit(){
    console.log("login btn clicked");

    if(this.loginData.username.trim() == "" || this.loginData.username == null){
      this._snack.open('Username is required !!', "", {
        duration : 3000,
        verticalPosition: 'top',
      });
    }
  }
}

```

```

return;
}

if(this.loginData.password.trim() == "" || this.loginData.password == null){
  this._snack.open('Password is required !!', "", {
    duration : 3000,
    verticalPosition: 'top',
  });
  return;
}

this.login.generateToken(this.loginData).subscribe(
  (data : any) => {
    //success
    console.log(data);
    console.log('success');
    //alert('success');
    //Swal.fire('Successfully done','User is registered and id is ' + data.id,'success');
    this.login.loginUser(data.token);
    this.login.getCurrentUser().subscribe((user:any) => {
      this.login.setUser(user);
      console.log(user);
      //redirect ....ADMIN : admin-dashboard
      //redirect ....user : user-dashboard

      if(this.login.getUserRole() == 'ADMIN'){
        //admin dashboard
        //window.location.href = '/admin';
        this.router.navigate(['admin']);
        this.login.loginStatusSubject.next(true);
      }else if(this.login.getUserRole() == 'NORMAL'){
        //user-dashboard
        //window.location.href = '/user-dashboard';
        this.router.navigate(['user-dashboard']);
        this.login.loginStatusSubject.next(true);
      }else{
        this.login.logout();
      }
    })

  });

},
(error) => {
  //error
  console.log(error);
  console.log('error');
  this._snack.open('Invalid credentials, Try again','',{

```

```

        duration : 3000,
        verticalPosition: 'top',
    });
    //alert('Something went wrong')
}
);

}
}

```

signup.component.html

```

<div class = "bootstrap-wrapper">
  <div class="container col-md-6 offset-md-3">
    <div class="row " style="margin-top: 20px">
      <div class="">
        <mat-card>
          <div class="container text-center">
            
          </div>
          <h1 class="text-center">Signup Here!!</h1>

          <!-- {{ user | json }} -->

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

            <mat-form-field class = "full-width "appearance="outline">
              <mat-label>Username</mat-label>
              <input
                required
                [(ngModel)] = "user.username"
                name = "username"
                matInput placeholder="Enter name"
              />
              <mat-hint>Username must be Unique!!</mat-hint>
            </mat-form-field>

            <mat-form-field class = "full-width "appearance="outline">
              <mat-label>Password</mat-label>
              <input
                required
                [(ngModel)] = "user.password"
                name = "password"
                type = "password" matInput placeholder="Enter password">
            </mat-form-field>

```

```
<mat-form-field class = "full-width "appearance="outline">
  <mat-label>First Name</mat-label>
  <input
    required
    [(ngModel)] = "user.firstName"
    name = "firstName"
    matInput placeholder="Enter first name">
</mat-form-field>

<mat-form-field class = "full-width "appearance="outline">
  <mat-label>Last Name</mat-label>
  <input
    required
    [(ngModel)] = "user.lastName"
    name = "lastName"
    matInput placeholder="Enter last name"/>
</mat-form-field>

<mat-form-field class = "full-width "appearance="outline">
  <mat-label>Email</mat-label>
  <input
    required
    [(ngModel)] = "user.email"
    name = "email"
    type = "email" matInput placeholder="Enter email">
</mat-form-field>

<mat-form-field class = "full-width "appearance="outline">
  <mat-label>Contact no.</mat-label>
  <input
    required
    [(ngModel)] = "user.phone"
    name = "phone"
    type = "number" matInput placeholder="Enter contact no.">
</mat-form-field>

<div class="container text-center">
  <button mat-raised-button type = "submit" color="primary">Sign Up</button>
  <button style="margin-left: 20px" mat-raised-button color="accent">Reset</button>
</div>
</form>
</mat-card>
</div>
</div>
</div>
```

signup.component.ts

```
import { Component, OnInit } from '@angular/core';
import { MatSnackBar } from '@angular/material/snack-bar';
import { UserService } from 'src/app/service/user.service';
import Swal from 'sweetalert2'

@Component({
  selector: 'app-signup',
  templateUrl: './signup.component.html',
  styleUrls: ['./signup.component.css']
})
export class SignupComponent implements OnInit {

  constructor(private userService:UserService, private _snack : MatSnackBar) { }

  public user = {
    username: "",
    password: "",
    firstName: "",
    lastName:"",
    email:"",
    phone:"",
  };

  ngOnInit(): void {
  }

  formSubmit(){
    console.log(this.user);

    this.Validation();

    //addUser:userService
    this.userService.addUser(this.user).subscribe(
      (data:any) => {
        //success
        console.log(data);
        //alert('success');
        Swal.fire('Successfully done','User is registered and id is ' + data.id,'success');

      },
      (error) => {
        //error
        console.log(error);
        this._snack.open('Something went wrong',"",{
```

```
        duration : 3000,  
        verticalPosition: 'top',  
    })  
    //alert('Something went wrong')  
  }  
)  
}
```

```
Validation(){
```

```
  if(this.user.username == "" || this.user.username == null){  
    this._snack.open('Username is required !!', "", {  
      duration : 3000,  
      verticalPosition: 'top',  
    });  
    return;  
  }
```

```
  if(this.user.password == "" || this.user.password == null){  
    this._snack.open('Password is required !!', "", {  
      duration : 3000,  
      verticalPosition: 'top',  
    });  
    return;  
  }
```

```
  if(this.user.firstName == "" || this.user.firstName == null){  
    this._snack.open('First Name is required !!', "", {  
      duration : 3000,  
      verticalPosition: 'top',  
    });  
    return;  
  }
```

```
  if(this.user.lastName == "" || this.user.lastName == null){  
    this._snack.open('Last Name is required !!', "", {  
      duration : 3000,  
      verticalPosition: 'top',  
    });  
    return;  
  }
```

```
  if(this.user.email == "" || this.user.email == null){  
    this._snack.open('Email is required !!', "", {  
      duration : 3000,  
      verticalPosition: 'top',  
    });  
    return;  
  }
```

```

    }

    if(this.user.phone == " " || this.user.phone == null){
        this._snack.open('Phone Number is required !!', "", {
            duration : 3000,
            verticalPosition: 'top',
        });
        return;
    }

}

}

```

payment.component.html

```

-----
<link href="//netdna.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css" rel="stylesheet"
id="bootstrap-css">
<script src="//netdna.bootstrapcdn.com/bootstrap/3.2.0/js/bootstrap.min.js"></script>
<script src="//code.jquery.com/jquery-1.11.1.min.js"></script>
<!-- Include the above in your HEAD tag ----->

<div class="container wrapper">
    <div class="row cart-head">
        <div class="container">
            <div class="row">
                <p></p>
            </div>
            <!-- <div class="row">
                <div style="display: table; margin: auto;">
                    <span class="step step_complete"> <a href="#" class="check-bc">Cart</a> <span
class="step_line step_complete"> </span> <span class="step_line backline"> </span> </span>
                    <span class="step step_complete"> <a href="#" class="check-bc">Checkout</a> <span
class="step_line "> </span> <span class="step_line step_complete"> </span> </span>
                    <span class="step_thankyou check-bc step_complete">Thank you</span>
                </div>
            </div> -->
            <div class="row">
                <p></p>
            </div>
        </div>
    </div>
    <div class="row cart-body">
        <form class="form-horizontal" method="post" action="">
            <div class="col-lg-6 col-md-6 col-sm-6 col-xs-12 col-md-push-6 col-sm-push-6">
                <!-- REVIEW ORDER-->
                <div class="panel panel-info">

```

```

<div class="panel-heading">
  Review Order <div class="pull-right"><small><a class="afix-1" href="#">Edit
Cart</a></small>
</div>
</div>
<div class="panel-body">
  <div class="form-group" *ngFor="let item of cartItems">
    <div class="col-sm-3 col-xs-3">
      <div mat-card-avatar class="example-header-image"></div>
    </div>

    <div class="col-sm-6 col-xs-6">
      <div class="col-xs-12">Product Name: {{ item.productName }}</div>
      <div class="col-xs-12"><small>Quantity:<span>{{ item.qty
}}</span></small></div>
    </div>
    <div class="col-sm-3 col-xs-3 text-right">
      <h6><span></span>{{ item.price | currency : 'INR' }}</h6>
    </div>
  </div>
  <div class="form-group">
    <hr />
  </div>
  <!-- <div class="form-group">
    <div class="col-sm-3 col-xs-3">
      
    </div>
    <div class="col-sm-6 col-xs-6">
      <div class="col-xs-12">Product name</div>
      <div class="col-xs-12"><small>Quantity:<span>1</span></small></div>
    </div>
    <div class="col-sm-3 col-xs-3 text-right">
      <h6><span>$</span>{{ item.price }}</h6>
    </div>
  </div>
  <div class="form-group">
    <hr />
  </div>
  <div class="form-group">
    <div class="col-sm-3 col-xs-3">
      
    </div>
    <div class="col-sm-6 col-xs-6">
      <div class="col-xs-12">Product name</div>
      <div class="col-xs-12"><small>Quantity:<span>2</span></small></div>
    </div>
  </div>

```

```

        <div class="col-sm-3 col-xs-3 text-right">
            <h6><span>$</span>50.00</h6>
        </div>
    </div> -->
    <div class="form-group">
        <hr />
    </div>
    <div class="form-group">
        <div class="col-xs-12">
            <strong>Subtotal</strong>
            <div class="pull-right"><span>{{ cartTotal | currency : 'INR' }}</span></div>
        </div>
        <div class="col-xs-12">
            <small>Shipping</small>
            <div class="pull-right"><span>Free</span></div>
        </div>
    </div>
    <div class="form-group">
        <hr />
    </div>
    <div class="form-group">
        <div class="col-xs-12">
            <strong>Order Total</strong>
            <div class="pull-right"><span>{{ cartTotal | currency : 'INR' }}</span></div>
        </div>
    </div>
</div>
<!--REVIEW ORDER END-->
</div>
<div class="col-lg-6 col-md-6 col-sm-6 col-xs-12 col-md-pull-6 col-sm-pull-6">
    <!--SHIPPING METHOD-->
    <div class="panel panel-info">
        <div class="panel-heading">Address</div>
        <div class="panel-body">
            <div class="form-group">
                <div class="col-md-12">
                    <h4>Shipping Address</h4>
                </div>
            </div>
            <div class="form-group">
                <div class="col-md-6 col-xs-12">
                    <strong>First Name:</strong>
                    <input type="text" name="first_name" class="form-control" value="" />
                </div>
                <div class="span1"></div>
                <div class="col-md-6 col-xs-12">
                    <strong>Last Name:</strong>

```

```

        <input type="text" name="last_name" class="form-control" value="" />
    </div>
</div>
<div class="form-group">
    <div class="col-md-12"><strong>Address:</strong></div>
    <div class="col-md-12">
        <input type="text" name="address" class="form-control" value="" />
    </div>
</div>
<div class="form-group">
    <div class="col-md-6 col-xs-12">
        <strong>City:</strong>
        <input type="text" name="city" class="form-control" value="" />
    </div>
    <div class="span1"></div>
    <div class="col-md-6 col-xs-12">
        <strong>State:</strong>
        <input type="text" name="state" class="form-control" value="" />
    </div>
</div>
<div class="form-group">
    <div class="col-md-12"><strong>Zip / Postal Code:</strong></div>
    <div class="col-md-12">
        <input type="text" name="zip_code" class="form-control" value="" />
    </div>
</div>
<div class="form-group">
    <div class="col-md-12"><strong>Phone Number:</strong></div>
    <div class="col-md-12"><input type="text" name="phone_number" class="form-
control"
        value="" />
    </div>
</div>
<div class="form-group">
    <div class="col-md-12"><strong>Email Address:</strong></div>
    <div class="col-md-12"><input type="text" name="email_address" class="form-
control"
        value="" />
    </div>
</div>
</div>
<!--SHIPPING METHOD END-->
<!--CREDIT CART PAYMENT-->
<div class="panel panel-info">
    <div class="panel-heading"><span><i class="glyphicon glyphicon-lock"></i></span>
Secure Payment
    </div>

```

```

<div class="panel-body">
  <div class="form-group">
    <div class="col-md-12"><strong>Card Type:</strong></div>
    <div class="col-md-12">
      <select id="CreditCardType" name="CreditCardType" class="form-control">
        <option value="5">Visa</option>
        <option value="6">MasterCard</option>
        <option value="7">American Express</option>
        <option value="8">Discover</option>
      </select>
    </div>
  </div>
  <div class="form-group">
    <div class="col-md-12"><strong>Credit Card Number:</strong></div>
    <div class="col-md-12"><input type="text" class="form-control"
name="car_number" value="" />
    </div>
  </div>
  <div class="form-group">
    <div class="col-md-12"><strong>Card CVV:</strong></div>
    <div class="col-md-12"><input type="text" class="form-control" name="car_code"
value="" />
    </div>
  </div>
  <div class="form-group">
    <div class="col-md-12">
      <strong>Expiration Date</strong>
    </div>
    <div class="col-lg-6 col-md-6 col-sm-6 col-xs-12">
      <select class="form-control" name="">
        <option value="">Month</option>
        <option value="01">01</option>
        <option value="02">02</option>
        <option value="03">03</option>
        <option value="04">04</option>
        <option value="05">05</option>
        <option value="06">06</option>
        <option value="07">07</option>
        <option value="08">08</option>
        <option value="09">09</option>
        <option value="10">10</option>
        <option value="11">11</option>
        <option value="12">12</option>
      </select>
    </div>
    <div class="col-lg-6 col-md-6 col-sm-6 col-xs-12">
      <select class="form-control" name="">
        <option value="">Year</option>

```

```

        <option value="2015">2015</option>
        <option value="2016">2016</option>
        <option value="2017">2017</option>
        <option value="2018">2018</option>
        <option value="2019">2019</option>
        <option value="2020">2020</option>
        <option value="2021">2021</option>
        <option value="2022">2022</option>
        <option value="2023">2023</option>
        <option value="2024">2024</option>
        <option value="2025">2025</option>
    </select>
</div>
</div>
<div class="form-group">
    <div class="col-md-12">
        <span>Pay secure using your credit card.</span>
    </div>
    <div class="col-md-12">
        <ul class="cards">
            <li class="visa hand">Visa</li>
            <li class="mastercard hand">MasterCard</li>
            <li class="amex hand">Amex</li>
        </ul>
        <div class="clearfix"></div>
    </div>
</div>
<div class="form-group">
    <div class="col-md-6 col-sm-6 col-xs-12">
        <button type="submit" class="btn btn-primary btn-submit-fix"
            (click)="PaymentDone()">Place
            Order</button>
    </div>
</div>
</div>
</div>
<!--CREDIT CART PAYMENT END-->
</div>

</form>
</div>
<div class="row cart-footer">

</div>
</div>

```

payment.component.ts

```
import { CartItem } from './../../../../model/cart-item';
import { MessengerService } from 'src/app/service/messenger.service';
import { Component, OnInit } from '@angular/core';
import { MatSnackBar } from '@angular/material/snack-bar';
import { ActivatedRoute, Router } from '@angular/router';
import Swal from 'sweetalert2';

@Component({
  selector: 'app-payment',
  templateUrl: './payment.component.html',
  styleUrls: ['./payment.component.css']
})
export class PaymentComponent implements OnInit {

  cartItems: [
    {id:1, productId: 2,productName:'Test1', qty: 3, price:344},
    {id:2, productId: 4, productName:'Test1', qty: 3, price:344},
    {id:3, productId: 5, productName:'Test1', qty: 3, price:344},
    {id:4, productId: 6, productName:'Test1',qty: 3, price:344},
  ];

  public payment = {
    name: "",
    address: "",
    phone: "",
    cardNumber: "",
    expDate: "",
    nameOnCard: ""
  }

  cartTotal;

  constructor(private _route: ActivatedRoute, private _router: Router, private _snack :
MatSnackBar, private msg: MessengerService) { }

  ngOnInit(): void {

    this.msg.sharedParam.subscribe((param: any)=>{
      console.log(param),
      this.cartItems = param}
    )

    this.msg.sharedParam2.subscribe((param: any)=>{
      console.log(param),
      this.cartTotal = param}
  )
}
```

```

    )
  }
  PaymentDone(){
    Swal.fire('Successfully done', 'Payment Successful!!', 'success');
  }
}

```

add-medicine.component.html

```

<mat-card>
  <h1>Add Medicine</h1>
  <div class="container">
    <div class="row">
      <div class="col-md-8 offset-md-2">
        <form (ngSubmit)= "addMedicine()">

          <mat-form-field class="w100" appearance="fill">
            <mat-label>Enter Name</mat-label>
            <input [(ngModel)]= "medicine.name" required type="text" name="name"
placeholder="Enter name here" matInput />
          </mat-form-field>

          <mat-form-field class="w100" appearance="fill">
            <mat-label>Enter Seller</mat-label>
            <input [(ngModel)]= "medicine.seller" required type="text" name="seller"
placeholder="Enter seller here" matInput />
          </mat-form-field>

          <mat-form-field class="w100" appearance="fill">
            <mat-label>Enter Description</mat-label>
            <textarea [(ngModel)]= "medicine.prodDesc" required name="prodDesc"
placeholder="Enter description here" matInput rows='5'></textarea>
          </mat-form-field>
          <div class="row">
            <div class="col-md-6">
              <mat-form-field class="w100" appearance="fill">
                <mat-label>Enter Offers</mat-label>
                <input [(ngModel)]= "medicine.offers" required type="text" name="offers"
placeholder="Enter offer here" matInput />
              </mat-form-field>

            </div>
            <div class="col-md-6">
              <mat-form-field class="w100" appearance="fill">
                <mat-label>Enter Price</mat-label>

```

```

        <input [(ngModel)]="medicine.price" required type="text" name="price"
placeholder="Enter price here" matInput />
    </mat-form-field>
</div>
</div>
<mat-form-field appearance="fill" class="w100">
    <mat-label>category</mat-label>
    <mat-select required name="category" [(ngModel)]="medicine.category.cid">
        <mat-option *ngFor="let c of categories" [value]="c.cid">
            {{c.title}}
        </mat-option>
    </mat-select>
</mat-form-field>
<br>
<mat-slide-toggle [(ngModel)]="medicine.active" name="active" class="mt10"
color="primary">
    Active status
</mat-slide-toggle>

<div class="container text-center">
    <button mat-raised-button color="accent" type="submit">Add</button>
</div>

</form>
</div>
</div>
</div>
</mat-card>

```

add-medicine.component.ts

```

import { Component, OnInit } from '@angular/core';
import { MatSnackBar } from '@angular/material/snack-bar';
import { CategoryService } from 'src/app/service/category.service';
import { MedicineService } from 'src/app/service/medicine.service';
import Swal from 'sweetalert2';

```

```

@Component({
  selector: 'app-add-medicine',
  templateUrl: './add-medicine.component.html',
  styleUrls: ['./add-medicine.component.css']
})
export class AddMedicineComponent implements OnInit {

```

```

  categories =[
  {

```

```

        cid:",
        title:"
    },
];

medicine = {
    id:",
    name:",
    seller:",
    prodDesc:",
    offers:",
    price:",
    active: true,
    category: {
        cid:",
    },
};

constructor(private categoryService: CategoryService, private medicineService: MedicineService,
private _snack: MatSnackBar) { }

ngOnInit(): void {
    this.categoryService.categories().subscribe(
        (data: any) => {
            //success
            //alert('success');
            this.categories = data;
            console.log(this.categories);
        },
        (error) => {
            //error
            console.log(error);
            Swal.fire('Error occur', 'Server error !!', 'error');

            //alert('Something went wrong')
        })
    }
}

```

```

addMedicine() {
  console.log(this.medicine);

  if (this.medicine.name.trim() == " || this.medicine.name == null) {
    this._snack.open('Name required !!', "", {
      duration: 3000,
      verticalPosition: 'top',
    });
    return;
  }
  if (this.medicine.seller.trim() == " || this.medicine.seller == null) {
    this._snack.open('Seller required !!', "", {
      duration: 3000,
      verticalPosition: 'top',
    });
    return;
  }
  if (this.medicine.prodDesc.trim() == " || this.medicine.prodDesc == null) {
    this._snack.open('Description required !!', "", {
      duration: 3000,
      verticalPosition: 'top',
    });
    return;
  }
  if (this.medicine.offers.trim() == " || this.medicine.offers == null) {
    this._snack.open('Offer required !!', "", {
      duration: 3000,
      verticalPosition: 'top',
    });
    return;
  }
  if (this.medicine.price.trim() == " || this.medicine.price == null) {
    this._snack.open('Price required !!', "", {
      duration: 3000,
      verticalPosition: 'top',
    });
    return;
  }
  // if (this.medicine.category..cidtrim() == " || this.medicine.category == null) {
  //   this._snack.open('Category required !!', "", {
  //     duration: 3000,
  //     verticalPosition: 'top',
  //   });
  //   return;
  // }
}

```

```

this.medicineService.addMedicine(this.medicine).subscribe(
  (data: any) => {
    //success
    console.log(data);
    this.medicine = {
      id:"",
      name:"",
      seller:"",
      prodDesc:"",
      offers:"",
      price:"",
      active: false,
      category: {
        cid:"",
      },
    };

    Swal.fire('Successfully done', 'Medicine is added ', 'success');
  },
  (error) => {
    //error
    console.log(error);
    Swal.fire('Error occur', 'Server error !!', 'error');

    //alert('Something went wrong')
  })
}
}

```

package.json

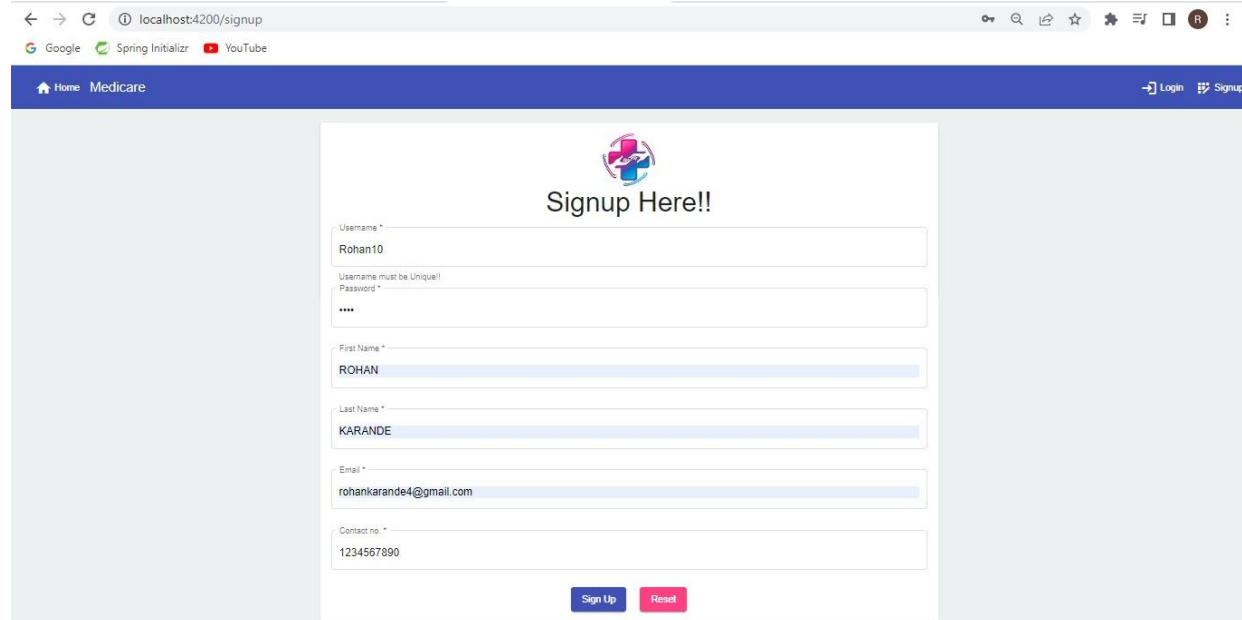
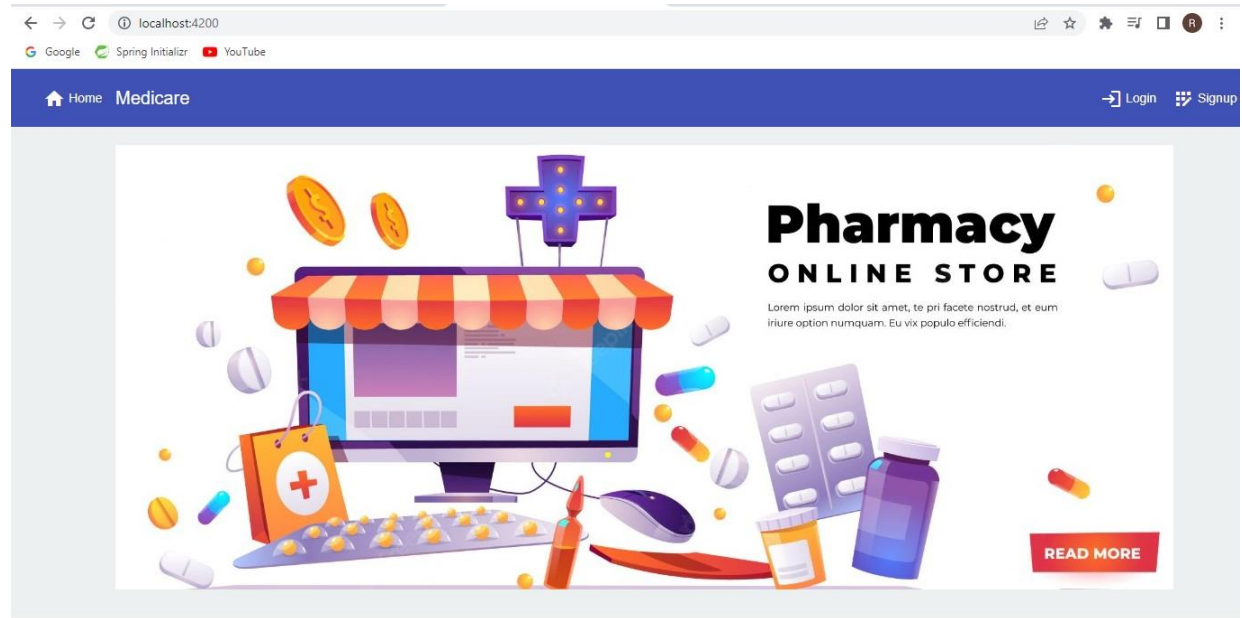
```

-----
{
  "name": "medicare-frontend",
  "version": "0.0.0",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build",
    "watch": "ng build --watch --configuration development",
    "test": "ng test"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "~12.2.0",
    "@angular/cdk": "^12.2.8",

```

```
"@angular/common": "~12.2.0",
"@angular/compiler": "~12.2.0",
"@angular/core": "~12.2.0",
"@angular/forms": "~12.2.0",
"@angular/material": "^12.2.8",
"@angular/platform-browser": "~12.2.0",
"@angular/platform-browser-dynamic": "~12.2.0",
"@angular/router": "~12.2.0",
"@ckeditor/ckeditor5-angular": "^2.0.2",
"@ckeditor/ckeditor5-build-classic": "^30.0.0",
"bootstrap-grid-only-css": "^4.1.3",
"ngx-ui-loader": "^11.0.0",
"rxjs": "~6.6.0",
"sweetalert2": "^11.1.7",
"tslib": "^2.3.0",
"zone.js": "~0.11.4"
},
"devDependencies": {
  "@angular-devkit/build-angular": "^12.2.11",
  "@angular/cli": "^12.2.7",
  "@angular/compiler-cli": "~12.2.0",
  "@types/jasmine": "~3.8.0",
  "@types/node": "^12.11.1",
  "jasmine-core": "~3.8.0",
  "karma": "~6.3.0",
  "karma-chrome-launcher": "~3.1.0",
  "karma-coverage": "~2.0.3",
  "karma-jasmine": "~4.0.0",
  "karma-jasmine-html-reporter": "~1.7.0",
  "typescript": "~4.3.5"
}
}
```


SCREENSHOTS -



← → ↻ localhost:4200/login

Google Spring Initializr YouTube

Home Medicare Login Signup



Login Here!!

Username *

Rohan10

Password *

Login Reset


← → ↻ localhost:4200/admin

Google Spring Initializr YouTube

Home Medicare Admin Logout

Menu

- Home
- Profile
- Category
- + Add Category
- + Medicines
- + Add Medicines



WELCOME ADMIN

← → ↻

localhost:4200/admin/add-category

🔑 🔍 ⚙️ 📱 🌐

Google Spring Initializr YouTube

Home MedicareAdmin Logout

Menu

- Home
- Profile
- Category
- Add Category
- Medicines
- Add Medicines

Add New Category

Title *

GENERAL

Description *

Contains generic medicines of day-to-day use.

Add

← → ↻

localhost:4200/admin/add-medicine

🔑 🔍 ⚙️ 📱 🌐

Google Spring Initializr YouTube

Home MedicareAdmin Logout

Menu

- Home
- Profile
- Category
- Add Category
- Medicines
- Add Medicines

Add Medicine

Enter Name *

DOLO650

Enter Seller *

Rakesh Chemist

Enter Description *

Generic purpose medicine for flu,cough etc.

Enter Offers *

10% OFF

Enter Price *

55/-

category *

GENERAL

☒ Active status

Add

