**Online Collaboration**

Developed By

Name: ARPAN BHADRA Reg. No: R201190200105

# NIIT

***Online Collaboration***

Batch Code : S210167

Start Date : 01.01.2021

End Date : 15.03.2021

Name of the Coordinator : LOPAMUDRA BERA

Name of the Developer : ARPAN BHADRA

Date of Submission : 15.03.2021

**NIIT**

###### CERTIFICATE

This is to certify that this report, titled **Online Collaboration** embodies the original work done by **Mr. ARPAN BHADRA**, in partial fulfilment of his course requirement at NIIT.

Coordinator : LOPAMUDRA BERA

### ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my teacher **LOPAMUDRA BERA** as well as to the education institute **NIIT** who gave me the golden opportunity to do this wonderful project on the topic **ONLINE COLLABORATION SERVICES**, which also helped me in doing a lot of Research and I came to know about so many new things I am really thankful to them.  
Secondly, I would also like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.

### ABSTRACT

This project manages the entire process of allocating project to existing employees as per his/her skillset working in that company

It’s also provides single window system to Employee, HR and Project Managers of a company to cater the skill specific requirements emerging in Projects.

### CONFIGURATION

Hardware:8GB RAM,2GB GRAPHICS CARD,1 TB HARD DISK, INTEL I3 8TH GENERATION PROCESSOR

Operating System: WINDOWS 10

Software:

* ECLIPSE 2019
  + - JAVA
    - Spring Boot
    - Hibernate
* MY SQL,
* Visual Studio Code
* Angular
* Html
* TypeScript
* SCSS
* JQuery

### TABLE OF CONTENTS

Chapter 1- Introduction

1.1 Objectives

1.2 Case Study

Chapter 2- Project Requirement Specification

2.1 Statement of Requirements

Chapter 3- Project Analysis

3.0 Project Plan

3.1 Display Model

3.2 Project Flow

3.3 Dispatcher Servlet Diagram

Chapter 4- Project Design

4.1 Angular Lifecycle Diagram

4.2 User Interface Design

4.3 Data Structure Model

4.4 Data Model

4.5 Project Structure

Chapter 5- Angular Framework

5.0 Front end

5.1 Personal Observation

### AIM

To create an application that operates via online recruiting website. Which stored the Records in Data Base and Also retrieving it.

### OBJECTIVES

This Project mainly use for the Professional Networking, allowing registered users to post blogs and their information is organize and easily access able to all the users.

Users will register themselves in the register form and fill there personal details. And Admin can able to manage all process from activating users to approving a blog posted by user.

User

* View Profile
* Update Profile
* Add Blog
* View Posted Blog

Admin

* View their own profile
* Update profie
* Activate or Deactivate user
* View All users
* Activate and Deactivate the blogs

LOGIN/REGISTRATION

**High level Case Study**

**My SQL**

**Database**

Back End

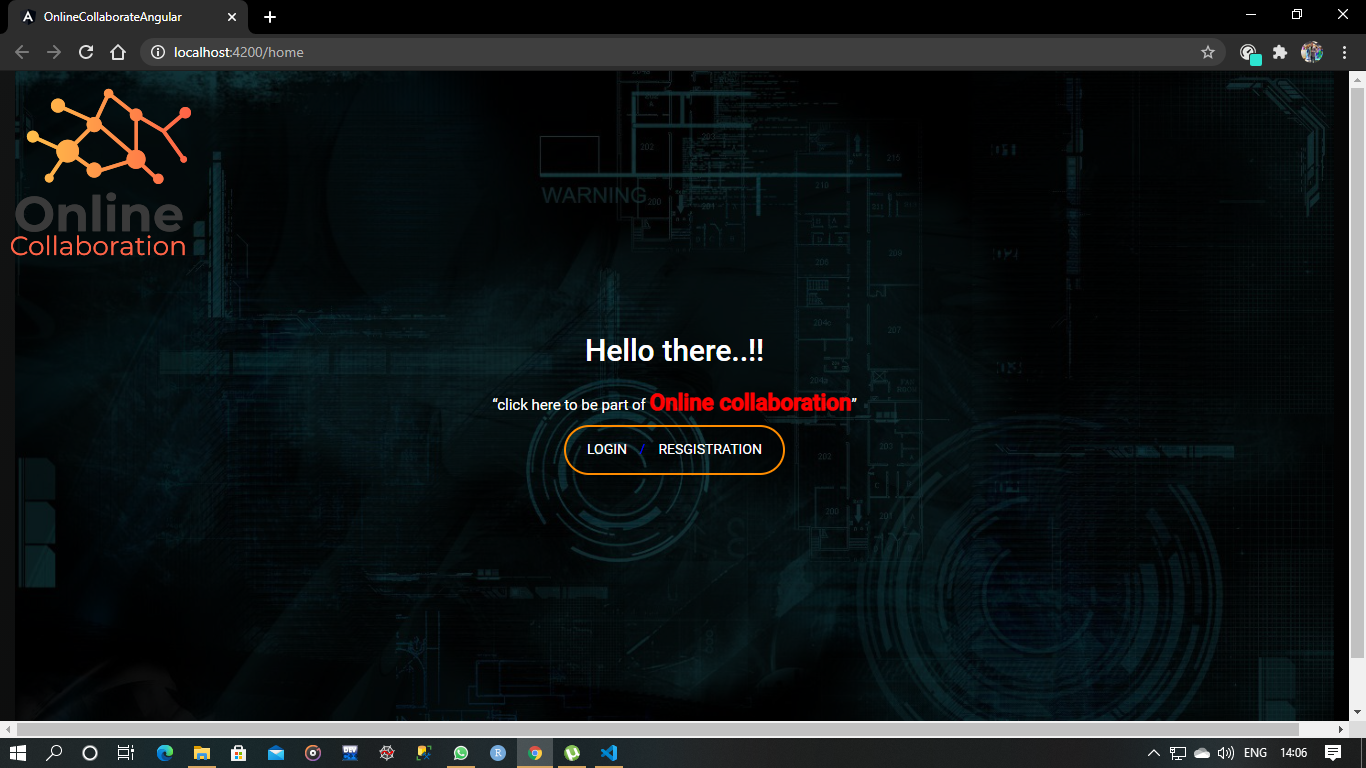
Sent Response

Sent Request

Admin

User

**User Interface Design**

****

**MODEL PACKAGES**

**USER MODEL:**

package com.coll.OnlineCollaborate.model;

import java.io.Serializable;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Transient;

import org.springframework.stereotype.Component;

@Component

@Entity

public class User extends DomainResponse implements Serializable {

private static final long serialVersionUID=1L;

@Id

@GeneratedValue(strategy=GenerationType.IDENTITY)

private int userId;

private String firstName;

private String lastName;

private String username;

private String password;

private String email;

private String role;

private String status;

private String isOnline;

private String enabled;

public int getUserId() {

return userId;

}

public void setUserId(int userId) {

this.userId = userId;

}

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 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 getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getRole() {

return role;

}

public void setRole(String role) {

this.role = role;

}

public String getStatus() {

return status;

}

public void setStatus(String status) {

this.status = status;

}

public String getIsOnline() {

return isOnline;

}

public void setIsOnline(String isOnline) {

this.isOnline = isOnline;

}

public String getEnabled() {

return enabled;

}

public void setEnabled(String enabled) {

this.enabled = enabled;

}

}

**BLOG MODEL:**

package com.coll.OnlineCollaborate.model;

import java.io.Serializable;

import java.time.LocalDate;

import java.util.List;

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 org.springframework.stereotype.Component;

import com.fasterxml.jackson.annotation.JsonManagedReference;

@Component

@Entity

public class Blog extends DomainResponse implements Serializable{

private static final long serialVersionUID=1L;

@Id

@GeneratedValue(strategy=GenerationType.IDENTITY)

int blogId;

String blogTitel, blogContent;

LocalDate blogPosted;

String status;

int noOfLikes,noFoComments, noOfViews;

int userId;

String username;

@OneToMany(mappedBy="blog", fetch=FetchType.EAGER,cascade=CascadeType.ALL)

@JsonManagedReference

List<BlogComments> blogComments;

public int getBlogId() {

return blogId;

}

public void setBlogId(int blogId) {

this.blogId = blogId;

}

public String getBlogTitel() {

return blogTitel;

}

public void setBlogTitel(String blogTitel) {

this.blogTitel = blogTitel;

}

public String getBlogContent() {

return blogContent;

}

public void setBlogContent(String blogContent) {

this.blogContent = blogContent;

}

public LocalDate getBlogPosted() {

return blogPosted;

}

public void setBlogPosted(LocalDate blogPosted) {

this.blogPosted = blogPosted;

}

public String getStatus() {

return status;

}

public void setStatus(String status) {

this.status = status;

}

public int getNoOfLikes() {

return noOfLikes;

}

public void setNoOfLikes(int noOfLikes) {

this.noOfLikes = noOfLikes;

}

public int getNoFoComments() {

return noFoComments;

}

public void setNoFoComments(int noFoComments) {

this.noFoComments = noFoComments;

}

public int getNoOfViews() {

return noOfViews;

}

public void setNoOfViews(int noOfViews) {

this.noOfViews = noOfViews;

}

public int getUserId() {

return userId;

}

public void setUserId(int userId) {

this.userId = userId;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public List<BlogComments> getBlogComments() {

return blogComments;

}

public void setBlogComments(List<BlogComments> blogComments) {

this.blogComments = blogComments;

}

}

**BLOGCOMMENTS MODEL:**

package com.coll.OnlineCollaborate.model;

import java.io.Serializable;

import java.time.LocalDate;

import java.util.Locale;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import org.springframework.stereotype.Component;

import com.fasterxml.jackson.annotation.JsonBackReference;

@Component

@Entity

public class BlogComments implements Serializable {

private static final long serialVersionUID=1L;

@Id

@GeneratedValue(strategy=GenerationType.IDENTITY)

int blogCommentId;

int userId;

String username;

String userProfileId;

String titel;

int onOfLikes;

String blogComment;

LocalDate currentDate;

@ManyToOne

@JoinColumn(name="BlogId")

Blog blog;

@JsonBackReference

public int getBlogCommentId() {

return blogCommentId;

}

public void setBlogCommentId(int blogCommentId) {

this.blogCommentId = blogCommentId;

}

public int getUserId() {

return userId;

}

public void setUserId(int userId) {

this.userId = userId;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getUserProfileId() {

return userProfileId;

}

public void setUserProfileId(String userProfileId) {

this.userProfileId = userProfileId;

}

public String getTitel() {

return titel;

}

public void setTitel(String titel) {

this.titel = titel;

}

public int getOnOfLikes() {

return onOfLikes;

}

public void setOnOfLikes(int onOfLikes) {

this.onOfLikes = onOfLikes;

}

public static long getSerialversionuid() {

return serialVersionUID;

}

public String getBlogComment() {

return blogComment;

}

public void setBlogComment(String blogComment) {

this.blogComment = blogComment;

}

public LocalDate getCurrentDate() {

return currentDate;

}

public void setCurrentDate(LocalDate currentDate) {

this.currentDate = currentDate;

}

public Blog getBlog() {

return blog;

}

public void setBlog(Blog blog) {

this.blog = blog;

}

}

**DATA ACCESS OBJECT MODEL**

**IUserDao:**

package com.coll.OnlineCollaborate.dao;

import java.util.List;

import com.coll.OnlineCollaborate.model.User;

public interface IUserDao {

List<User> userListbyStatus(String status);

List<User> getAllusers();

User getUserById(int userId);

User getUserByName(String username);

User validateUser(User user);

boolean addUser(User user);

boolean updateUser(User user);

boolean deleteUser(int userId);

boolean deactiveUser(int userId);

boolean activeUser(int userId);

boolean logoutUser(int userId);

List<User> getAllDeactiveUser();

boolean updateUserProfile(String file, Integer userId);

}

**IBlogDao:**

package com.coll.OnlineCollaborate.dao;

import java.util.List;

import com.coll.OnlineCollaborate.model.Blog;

public interface IBlogDao {

List<Blog> getAllBlogs();

List<Blog> getBlogsByStatus(String status);

List<Blog> getUsersBlogs(int id);

Blog getBlogById(int blogId);

boolean addBlog(Blog blog);

boolean updateBlog(Blog blog);

boolean deleteBlog(int blogId);

}

**IBlogCommentsDao:**

package com.coll.OnlineCollaborate.dao;

import java.util.List;

import com.coll.OnlineCollaborate.model.Blog;

import com.coll.OnlineCollaborate.model.BlogComments;

public interface IBlogCommentsDao {

List<BlogComments> getAllBlogComments();

List<BlogComments> getCommentsbyBlogId(int blogId);

BlogComments getBlogCommentsById(int blogCommentsId);

boolean addBlogComments(BlogComments blogComments);

boolean updateBlogComments(BlogComments blogComments);

boolean deleteBlogComments(int blogCommentsId);

}

**DAO IMPL**

**UserDaoImpl:**

package com.coll.OnlineCollaborate.daoImpl;

import java.util.List;

import org.hibernate.SessionFactory;

import org.hibernate.query.Query;

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

import org.springframework.stereotype.Repository;

import org.springframework.transaction.annotation.Transactional;

import com.coll.OnlineCollaborate.dao.IUserDao;

import com.coll.OnlineCollaborate.model.User;

@Repository("userDao")

@Transactional

public class UserDaoImpl implements IUserDao {

@Autowired

SessionFactory sessionFactory;

@Override

public List<User> userListbyStatus(String status) {

String q="from User where status="+status+"'";

Query query=sessionFactory.getCurrentSession().createQuery(q);

return query.getResultList();

}

@Override

public List<User> getAllusers() {

return sessionFactory.getCurrentSession().createQuery("from User",User.class).getResultList();

}

@Override

public User getUserById(int userId) {

return sessionFactory.getCurrentSession().get(User.class, Integer.valueOf(userId));

}

@Override

public User getUserByName(String username) {

String query="from User where username=:username";

return sessionFactory.getCurrentSession().createQuery(query,User.class).setParameter("username", username).getSingleResult();

}

@Override

public User validateUser(User user) {

String username=user.getUsername();

String password=user.getPassword();

String q="from User where username='"+username+"' and password='"+password+"' and enabled='true'";

Query query=sessionFactory.getCurrentSession().createQuery(q);

try {

user=(User)query.getSingleResult();

user.setIsOnline("true");

return user;

}

catch(Exception e) {

e.printStackTrace();

return null;

}

}

@Override

public boolean addUser(User user) {

try {

sessionFactory.getCurrentSession().save(user);

return true;

}

catch(Exception ex) {

ex.printStackTrace();

return false;

}

}

@Override

public boolean updateUser(User user) {

try {

sessionFactory.getCurrentSession().update(user);

return true;

}

catch(Exception ex) {

ex.printStackTrace();

return false;

}

}

@Override

public boolean deleteUser(int userId) {

try {

sessionFactory.getCurrentSession().delete(getUserById(userId));

return true;

}

catch(Exception ex) {

ex.printStackTrace();

return false;

}

}

@Override

public boolean deactiveUser(int userId) {

try {

User user=getUserById(userId);

user.setEnabled("false");

user.setStatus("Inactive");

sessionFactory.getCurrentSession().update(user);

return true;

}

catch(Exception ex) {

ex.printStackTrace();

return false;

}

}

@Override

public boolean updateUserProfile(String file, Integer userId) {

String q="update User set profile=:filename where userId=:id";

Query query=sessionFactory.getCurrentSession().createQuery(q);

query.setParameter("id",(Integer)userId);

query.setParameter("filename", file);

try {

query.executeUpdate();

return true;

}

catch(Exception e) {

e.printStackTrace();

return false;

}

}

@Override

public boolean activeUser(int userId) {

try {

User user=getUserById(userId);

user.setEnabled("true");

user.setStatus("Active");

sessionFactory.getCurrentSession().update(user);

return true;

}

catch(Exception ex) {

ex.printStackTrace();

return false;

}

}

@Override

public List<User> getAllDeactiveUser() {

return sessionFactory.getCurrentSession().createQuery("from User where enabled='false'",User.class).getResultList();

}

@Override

public boolean logoutUser(int userId) {

try {

User user=getUserById(userId);

user.setIsOnline("false");

sessionFactory.getCurrentSession().update(user);

return true;

}

catch(Exception ex) {

ex.printStackTrace();

return false;

}

}

}

**BlogDaoImpl:**

package com.coll.OnlineCollaborate.daoImpl;

import java.util.List;

import org.hibernate.SessionFactory;

import org.hibernate.query.Query;

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

import org.springframework.stereotype.Repository;

import org.springframework.transaction.annotation.Transactional;

import com.coll.OnlineCollaborate.dao.IBlogDao;

import com.coll.OnlineCollaborate.model.Blog;

import com.coll.OnlineCollaborate.model.User;

@Repository("blogDao")

@Transactional

public class BlogDaoImpl implements IBlogDao{

@Autowired

SessionFactory sessionFactory;

@Override

public List<Blog> getAllBlogs() {

return sessionFactory.getCurrentSession().createQuery("from Blog",Blog.class).getResultList();

}

@Override

public List<Blog> getBlogsByStatus(String status) {

String q="from Blog where status='"+status+"'";

Query query=sessionFactory.getCurrentSession().createQuery(q);

return query.getResultList();

}

@Override

public List<Blog> getUsersBlogs(int id) {

return sessionFactory.getCurrentSession().createQuery("from Blog",Blog.class).getResultList();

}

@Override

public Blog getBlogById(int blogId) {

return sessionFactory.getCurrentSession().get(Blog.class, Integer.valueOf(blogId));

}

@Override

public boolean addBlog(Blog blog) {

try {

sessionFactory.getCurrentSession().save(blog);

return true;

}

catch(Exception ex) {

ex.printStackTrace();

return false;

}

}

@Override

public boolean updateBlog(Blog blog) {

try {

sessionFactory.getCurrentSession().update(blog);

return true;

}

catch(Exception ex) {

ex.printStackTrace();

return false;

}

}

@Override

public boolean deleteBlog(int blogId) {

try {

sessionFactory.getCurrentSession().delete(getBlogById(blogId));

return true;

}

catch(Exception ex) {

ex.printStackTrace();

return false;

}

}

}

**BlogCommentsDaoImpl:**

package com.coll.OnlineCollaborate.daoImpl;

import java.util.List;

import org.hibernate.SessionFactory;

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

import org.springframework.stereotype.Repository;

import org.springframework.transaction.annotation.Transactional;

import com.coll.OnlineCollaborate.dao.IBlogCommentsDao;

import com.coll.OnlineCollaborate.model.Blog;

import com.coll.OnlineCollaborate.model.BlogComments;

@Repository("blogCommentsDao")

@Transactional

public class BlogCommentsDaoImpl implements IBlogCommentsDao {

@Autowired

SessionFactory sessionFactory;

@Override

public List<BlogComments> getAllBlogComments() {

return sessionFactory.getCurrentSession().createQuery("from BlogComments", BlogComments.class).getResultList();

}

@Override

public BlogComments getBlogCommentsById(int blogId) {

return sessionFactory.getCurrentSession().get(BlogComments.class, Integer.valueOf(blogId));

}

@Override

public boolean addBlogComments(BlogComments blogComments) {

try {

sessionFactory.getCurrentSession().save(blogComments);

return true;

}

catch(Exception ex) {

ex.printStackTrace();

return false;

}

}

@Override

public boolean updateBlogComments(BlogComments blogComments) {

try {

sessionFactory.getCurrentSession().update(blogComments);

return true;

}

catch(Exception ex) {

ex.printStackTrace();

return false;

}

}

@Override

public boolean deleteBlogComments(int blogCommentsId) {

try {

sessionFactory.getCurrentSession().delete(getBlogCommentsById(blogCommentsId));

return true;

}

catch(Exception ex) {

ex.printStackTrace();

return false;

}

}

@Override

public List<BlogComments> getCommentsbyBlogId(int blogId) {

return (List<BlogComments>) sessionFactory.getCurrentSession().get(BlogComments.class,Integer.valueOf(blogId));

}

}

**SERVICE PACKAGE**

**IUserService:**

package com.coll.OnlineCollaborate.service;

import java.util.List;

import com.coll.OnlineCollaborate.model.User;

public interface IUserService {

List<User> userListbyStatus(String status);

List<User> getAllusers();

User getUserById(int userId);

User getUserByName(String username);

User validateUser(User user);

boolean addUser(User user);

boolean updateUser(User user);

boolean deleteUser(int userId);

boolean deactiveUser(int userId);

boolean activeUser(int userId);

boolean logoutUser(int userId);

List<User> getAllDeactiveUser();

boolean updateUserProfile(String file, Integer userId);

}

**IBlogService:**

package com.coll.OnlineCollaborate.service;

import java.util.List;

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

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

import com.coll.OnlineCollaborate.model.Blog;

public interface IBlogService {

List<Blog> getAllBlogs();

List<Blog> getBlogsByStatus(String status);

List<Blog> getUsersBlogs(int id);

Blog getBlogById(int blogId);

boolean addBlog(Blog blog);

boolean updateBlog(Blog blog);

boolean deleteBlog(int blogId);

}

**IBlogCommentsService:**

package com.coll.OnlineCollaborate.service;

import java.util.List;

import com.coll.OnlineCollaborate.model.BlogComments;

public interface IBlogCommentsService {

List<BlogComments> getAllBlogComments();

List<BlogComments> getCommentsbyBlogId(int blogId);

BlogComments getBlogCommentsById(int blogCommentsId);

boolean addBlogComments(BlogComments blogComments);

boolean updateBlogComments(BlogComments blogComments);

boolean deleteBlogComments(int blogCommentsId);

}

**SERVICEIMPL PACKAGE**

**UserServiceImpl:**

package com.coll.OnlineCollaborate.serviceImpl;

import java.util.List;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.coll.OnlineCollaborate.dao.IUserDao;

import com.coll.OnlineCollaborate.model.User;

import com.coll.OnlineCollaborate.service.IUserService;

@Service

@Transactional

public class UserServiceImpl implements IUserService {

@Autowired

IUserDao userDao;

@Override

public List<User> userListbyStatus(String status) {

// TODO Auto-generated method stub

return userDao.userListbyStatus(status);

}

@Override

public List<User> getAllusers() {

// TODO Auto-generated method stub

return userDao.getAllusers();

}

@Override

public User getUserById(int userId) {

// TODO Auto-generated method stub

return userDao.getUserById(userId);

}

@Override

public User getUserByName(String username) {

// TODO Auto-generated method stub

return userDao.getUserByName(username);

}

@Override

public User validateUser(User user) {

// TODO Auto-generated method stub

return userDao.validateUser(user);

}

@Override

public boolean addUser(User user) {

// TODO Auto-generated method stub

return userDao.addUser(user);

}

@Override

public boolean updateUser(User user) {

// TODO Auto-generated method stub

return userDao.updateUser(user);

}

@Override

public boolean deleteUser(int userId) {

// TODO Auto-generated method stub

return userDao.deleteUser(userId);

}

@Override

public boolean deactiveUser(int userId) {

// TODO Auto-generated method stub

return userDao.deactiveUser(userId);

}

@Override

public boolean updateUserProfile(String file, Integer userId) {

// TODO Auto-generated method stub

return userDao.updateUserProfile(file, userId);

}

@Override

public boolean activeUser(int userId) {

// TODO Auto-generated method stub

return userDao.activeUser(userId);

}

@Override

public List<User> getAllDeactiveUser() {

// TODO Auto-generated method stub

return userDao.getAllDeactiveUser();

}

@Override

public boolean logoutUser(int userId) {

// TODO Auto-generated method stub

return userDao.logoutUser(userId);

}

}

**BlogServiceImpl:**

package com.coll.OnlineCollaborate.serviceImpl;

import java.util.List;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.coll.OnlineCollaborate.dao.IBlogDao;

import com.coll.OnlineCollaborate.model.Blog;

import com.coll.OnlineCollaborate.service.IBlogService;

@Service

@Transactional

public class BlogServiceImpl implements IBlogService {

@Autowired

IBlogDao blogDao;

@Override

public List<Blog> getAllBlogs() {

return blogDao.getAllBlogs();

}

@Override

public List<Blog> getBlogsByStatus(String status) {

return blogDao.getBlogsByStatus(status);

}

@Override

public List<Blog> getUsersBlogs(int blogId) {

return blogDao.getUsersBlogs(blogId);

}

@Override

public Blog getBlogById(int blogId) {

return blogDao.getBlogById(blogId);

}

@Override

public boolean addBlog(Blog blog) {

return blogDao.addBlog(blog);

}

@Override

public boolean updateBlog(Blog blog) {

return blogDao.updateBlog(blog);

}

@Override

public boolean deleteBlog(int blogId) {

return blogDao.deleteBlog(blogId);

}

}

**BlogCommentsServiceImpl:**

package com.coll.OnlineCollaborate.serviceImpl;

import java.util.List;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.coll.OnlineCollaborate.dao.IBlogCommentsDao;

import com.coll.OnlineCollaborate.model.BlogComments;

import com.coll.OnlineCollaborate.service.IBlogCommentsService;

@Service

@Transactional

public class BlogCommentsServiceImpl implements IBlogCommentsService {

@Autowired

IBlogCommentsDao blogCommentsDao;

@Override

public List<BlogComments> getAllBlogComments() {

return blogCommentsDao.getAllBlogComments();

}

@Override

public BlogComments getBlogCommentsById(int blogCommentsId) {

return blogCommentsDao.getBlogCommentsById(blogCommentsId);

}

@Override

public boolean addBlogComments(BlogComments blogComments) {

return blogCommentsDao.addBlogComments(blogComments);

}

@Override

public boolean updateBlogComments(BlogComments blogComments) {

return blogCommentsDao.updateBlogComments(blogComments);

}

@Override

public boolean deleteBlogComments(int blogCommentsId) {

return blogCommentsDao.deleteBlogComments(blogCommentsId);

}

@Override

public List<BlogComments> getCommentsbyBlogId(int blogId) {

return blogCommentsDao.getCommentsbyBlogId(blogId);

}

}

**CONTROLLER PACKAGE**

**UserController:**

package com.coll.OnlineCollaborate.controller;

import java.util.List;

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

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

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

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

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

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

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

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

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

import com.coll.OnlineCollaborate.model.User;

import com.coll.OnlineCollaborate.service.IUserService;

@RestController

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

@RequestMapping(value="/api")

public class UserController {

@Autowired

IUserService userService;

@PostMapping("save-user")public boolean saveUser(@RequestBody User user) {

return userService.addUser(user);

}

@GetMapping("user-list")

public List<User> allUsers(){

return userService.getAllusers();

}

@DeleteMapping("delete-user/{userId}")

public boolean deleteUser(@PathVariable("userId") int userId) {

return userService.deleteUser(userId);

}

@GetMapping("user/{userId}")

public User userById(@PathVariable("userId") int userId) {

return userService.getUserById(userId);

}

@PostMapping("update-user/{userId}")

public boolean updateUser(@RequestBody User user, @PathVariable("userId") int userId) {

user.setUserId(userId);

return userService.updateUser(user);

}

@GetMapping("deactive-list")

public List<User> AllDeactiveUser(){

return userService.getAllDeactiveUser();

}

@PostMapping("active-user/{userId}")

public boolean activeUser(@RequestBody User user, @PathVariable("userId") int userId) {

return userService.activeUser(userId);

}

@PostMapping("validate-user")

public User validateUser(@RequestBody User user) {

return userService.validateUser(user);

}

@PostMapping("logout-user/{userId}")

public boolean logoutUser(@RequestBody User user, @PathVariable("userId") int userId) {

return userService.logoutUser(userId);

}

}

**BlogController:**

package com.coll.OnlineCollaborate.controller;

import java.util.List;

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

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

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

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

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

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

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

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

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

import com.coll.OnlineCollaborate.model.Blog;

import com.coll.OnlineCollaborate.model.User;

import com.coll.OnlineCollaborate.service.IBlogService;

@RestController

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

@RequestMapping(value="/api")

public class BlogController {

@Autowired

IBlogService blogService;

@PostMapping("save-blog")

public boolean saveBlog(@RequestBody Blog blog)

{

return blogService.addBlog(blog);

}

@GetMapping("blog-list")

public List<Blog> allBlog()

{

return blogService.getAllBlogs();

}

@DeleteMapping("delete-blog/{blogId}")

public boolean deleteBlog(@PathVariable("blogId") int blogId) {

return blogService.deleteBlog(blogId);

}

@GetMapping("blog/{blogId}")

public Blog blogById(@PathVariable("blogId") int blogId)

{

return blogService.getBlogById(blogId);

}

@PostMapping("update-blog/{blog}")

public boolean updateBlog(@PathVariable("blog") Blog blog)

{

return blogService.updateBlog(blog);

}

}

**BlogCommentsController:**

package com.coll.OnlineCollaborate.controller;

import java.util.List;

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

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

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

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

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

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

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

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

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

import com.coll.OnlineCollaborate.model.Blog;

import com.coll.OnlineCollaborate.model.BlogComments;

import com.coll.OnlineCollaborate.service.IBlogCommentsService;

@RestController

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

@RequestMapping(value="/api")

public class BlogCommentsController {

@Autowired

IBlogCommentsService blogCommentsService;

@PostMapping("save-blogComments")

public boolean saveBlog(@RequestBody BlogComments blogComments)

{

return blogCommentsService.addBlogComments(blogComments);

}

@GetMapping("blogComments-list")

public List<BlogComments> allBlogComments()

{

return blogCommentsService.getAllBlogComments();

}

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

public boolean deleteBlogComments(@PathVariable("blogComments\_id") int blogComments\_id) {

return blogCommentsService.deleteBlogComments(blogComments\_id);

}

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

public BlogComments blogCommentsById(@PathVariable("blogComments\_id") int blogComments\_id)

{

return blogCommentsService.getBlogCommentsById(blogComments\_id);

}

@PostMapping("update-blogComments/{blogComments}")

public boolean updateBlogComments(@PathVariable("blogComments") BlogComments blogComments)

{

return blogCommentsService.updateBlogComments(blogComments);

}

}

**CONFIGURATION PACKAGE**

**HibernateConfig:**

package com.coll.OnlineCollaborate.config;

import java.util.Properties;

import javax.sql.DataSource;

import org.hibernate.SessionFactory;

import org.springframework.boot.autoconfigure.EnableAutoConfiguration;

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

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.context.annotation.ComponentScans;

import org.springframework.context.annotation.Configuration;

import org.springframework.jdbc.datasource.DriverManagerDataSource;

import org.springframework.orm.hibernate5.HibernateTransactionManager;

import org.springframework.orm.hibernate5.LocalSessionFactoryBean;

import org.springframework.orm.hibernate5.LocalSessionFactoryBuilder;

import org.springframework.transaction.annotation.EnableTransactionManagement;

import org.springframework.web.servlet.ViewResolver;

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

@Configuration

@ComponentScans(value= {@ComponentScan("com.coll.OnlineCollaboration"),

@ComponentScan("model"),

@ComponentScan("controller"),

@ComponentScan("dao"),

@ComponentScan("daoImpl"),

@ComponentScan("config"),

@ComponentScan("serviceImpl"),

@ComponentScan("service")})

@EnableAutoConfiguration(exclude= {HibernateJpaAutoConfiguration.class})

@EnableTransactionManagement

public class HibernateConfig {

public static final String DATABASE\_URL="jdbc:mysql://localhost:3306/collaboration";

public static final String DATABASE\_DRIVER="com.mysql.cj.jdbc.Driver";

public static final String DATABASE\_DIALECT="org.hibernate.dialect.MySQLDialect";

public static final String DATABASE\_USERNAME="root";

public static final String DATABASE\_PASSWORD="niit@123";

@Bean(name="datasource")

public DataSource getDataSource() {

DriverManagerDataSource datasource=new DriverManagerDataSource();

datasource.setDriverClassName(DATABASE\_DRIVER);

datasource.setUrl(DATABASE\_URL);

datasource.setUsername(DATABASE\_USERNAME);

datasource.setPassword(DATABASE\_PASSWORD);

return datasource;

}

@Bean

public LocalSessionFactoryBean getSessionFactory() {

LocalSessionFactoryBean sessionFactory=new LocalSessionFactoryBean();

sessionFactory.setDataSource(getDataSource());

sessionFactory.setPackagesToScan("com.coll.OnlineCollaborate");

Properties hibernateProperties=new Properties();

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

hibernateProperties.put("hibernate.show\_sql", "true");

hibernateProperties.put("hibernate.hbm2ddl.auto", "update");

sessionFactory.setHibernateProperties(hibernateProperties);

return sessionFactory;

}

@Bean

public HibernateTransactionManager getTranscationManager() {

HibernateTransactionManager txm=new HibernateTransactionManager();

txm.setSessionFactory(getSessionFactory().getObject());

return txm;

}

@Bean

public ViewResolver jspViewResolver()

{

InternalResourceViewResolver viewResolver=new InternalResourceViewResolver();

viewResolver.setPrefix("/views");

viewResolver.setSuffix(".jsp");

return viewResolver;

}

}

**Project:**

1. Create a SpringBoot project named OnlineCollaboration with web,

Spring Data JPA, SpringBoot Dev Tools and MySQL Server Driver packages. Extract that project.

1. Import the project in Eclipse.
2. Create the configuration class  
    Instead of XML, we perform annotation-based configuration. So, we create a class HibernateConfig.java and specify the required configuration in it. However, there is one more configuration class OnlineCollaborateApplication.java. This class is provided by Spring Boot automatically.

**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* [*https://maven.apache.org/xsd/maven-4.0.0.xsd*](https://maven.apache.org/xsd/maven-4.0.0.xsd)*"*>

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.4.2</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.coll</groupId>

<artifactId>OnlineCollaborate</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>OnlineCollaborate</name>

<description>Demo project for Spring Boot</description>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-mail</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-validation</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</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>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**HibernateConfig**

package com.coll.OnlineCollaborate.config;

import java.util.Properties;

import javax.sql.DataSource;

import org.springframework.boot.autoconfigure.EnableAutoConfiguration;

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

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.context.annotation.ComponentScans;

import org.springframework.context.annotation.Configuration;

import org.springframework.jdbc.datasource.DriverManagerDataSource;

import org.springframework.orm.hibernate5.HibernateTransactionManager;

import org.springframework.orm.hibernate5.LocalSessionFactoryBean;

import org.springframework.transaction.annotation.EnableTransactionManagement;

import org.springframework.web.servlet.ViewResolver;

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

@Configuration

@ComponentScans(value ={@ComponentScan("com.coll.OnlineCollaborate"),

@ComponentScan("model"),

@ComponentScan("controller"),

@ComponentScan("dao"),

@ComponentScan("daoImpl"),

@ComponentScan("config"),

@ComponentScan("service"),

@ComponentScan("serviceImpl")

})

@EnableAutoConfiguration(exclude = {HibernateJpaAutoConfiguration.class})

@EnableTransactionManagement

public class HibernateConfig {

public static final String DATABASE\_URL="jdbc:mysql://localhost:3306/collaboration";

public static final String DATABASE\_DRIVER="com.mysql.cj.jdbc.Driver";

public static final String DATABASE\_DIALECT="org.hibernate.dialect.MySQLDialect";

public static final String DATABASE\_USERNAME="root";

public static final String DATABASE\_PASSWORD="krishnanCse@76";

@Bean(name="datasource")

public DataSource getDataSource()

{

DriverManagerDataSource dataSource =new DriverManagerDataSource();

dataSource.setDriverClassName(DATABASE\_DRIVER);

dataSource.setUrl(DATABASE\_URL);

dataSource.setUsername(DATABASE\_USERNAME);

dataSource.setPassword(DATABASE\_PASSWORD);

return dataSource;

}

@Bean

public LocalSessionFactoryBean getSessionFactory()

{

LocalSessionFactoryBean sessionFactory=new LocalSessionFactoryBean();

sessionFactory.setDataSource(getDataSource());

sessionFactory.setPackagesToScan("com.coll.OnlineCollaborate");

Properties hibernateProperties=new Properties();

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

hibernateProperties.put("hibernate.show\_sql","true");

hibernateProperties.put("hibernate.hbm2dll.auto","update");

sessionFactory.setHibernateProperties(hibernateProperties);

return sessionFactory;

}

@Bean

public HibernateTransactionManager getTransactionManager()

{

HibernateTransactionManager txm=new HibernateTransactionManager();

txm.setSessionFactory(getSessionFactory().getObject());

return txm;

}

@Bean

public ViewResolver jspViewResolver()

{

InternalResourceViewResolver viewResolver=new InternalResourceViewResolver();

viewResolver.setPrefix("/views/");

viewResolver.setSuffix(".jsp");

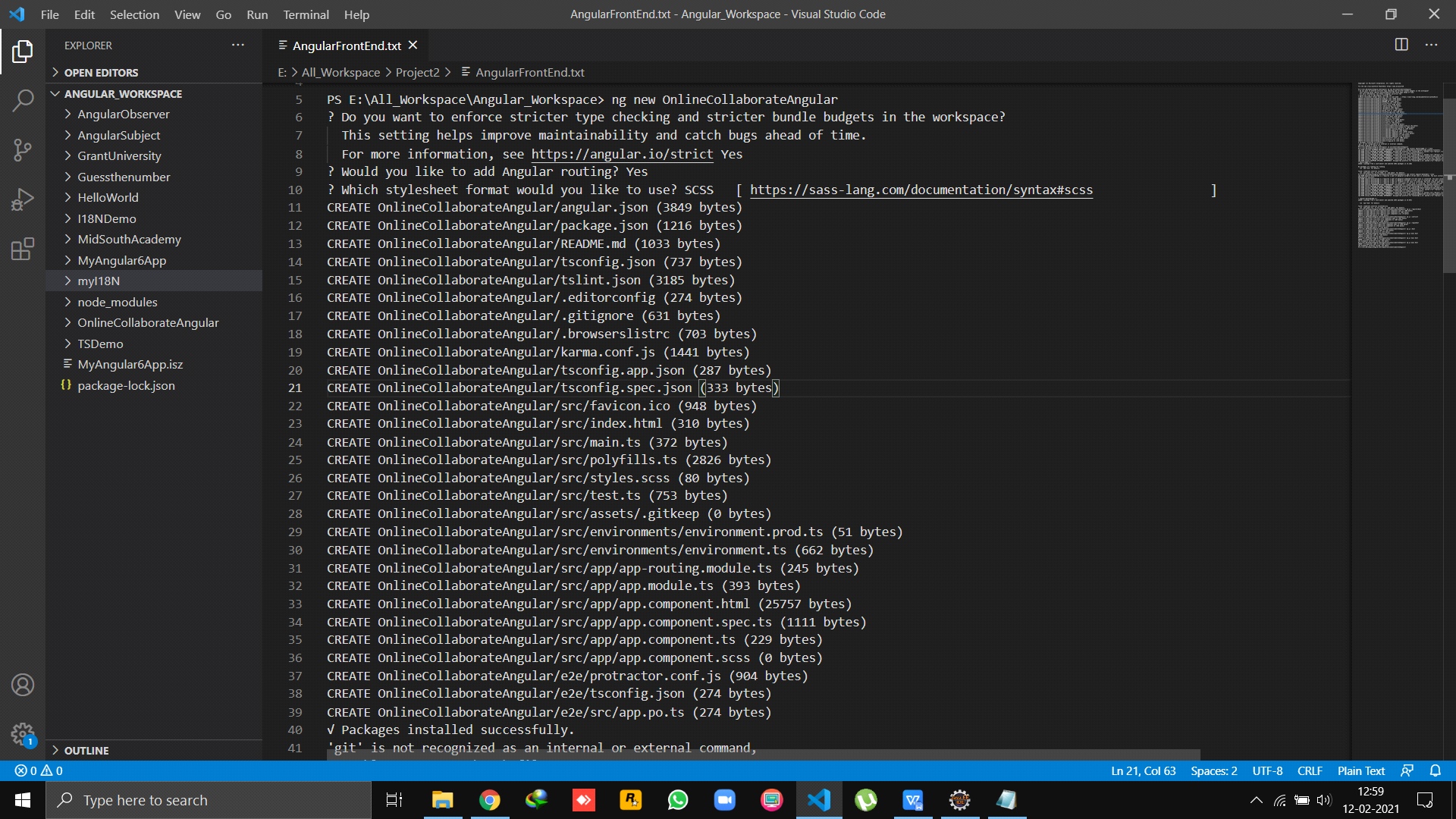
return viewResolver;

}

}

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

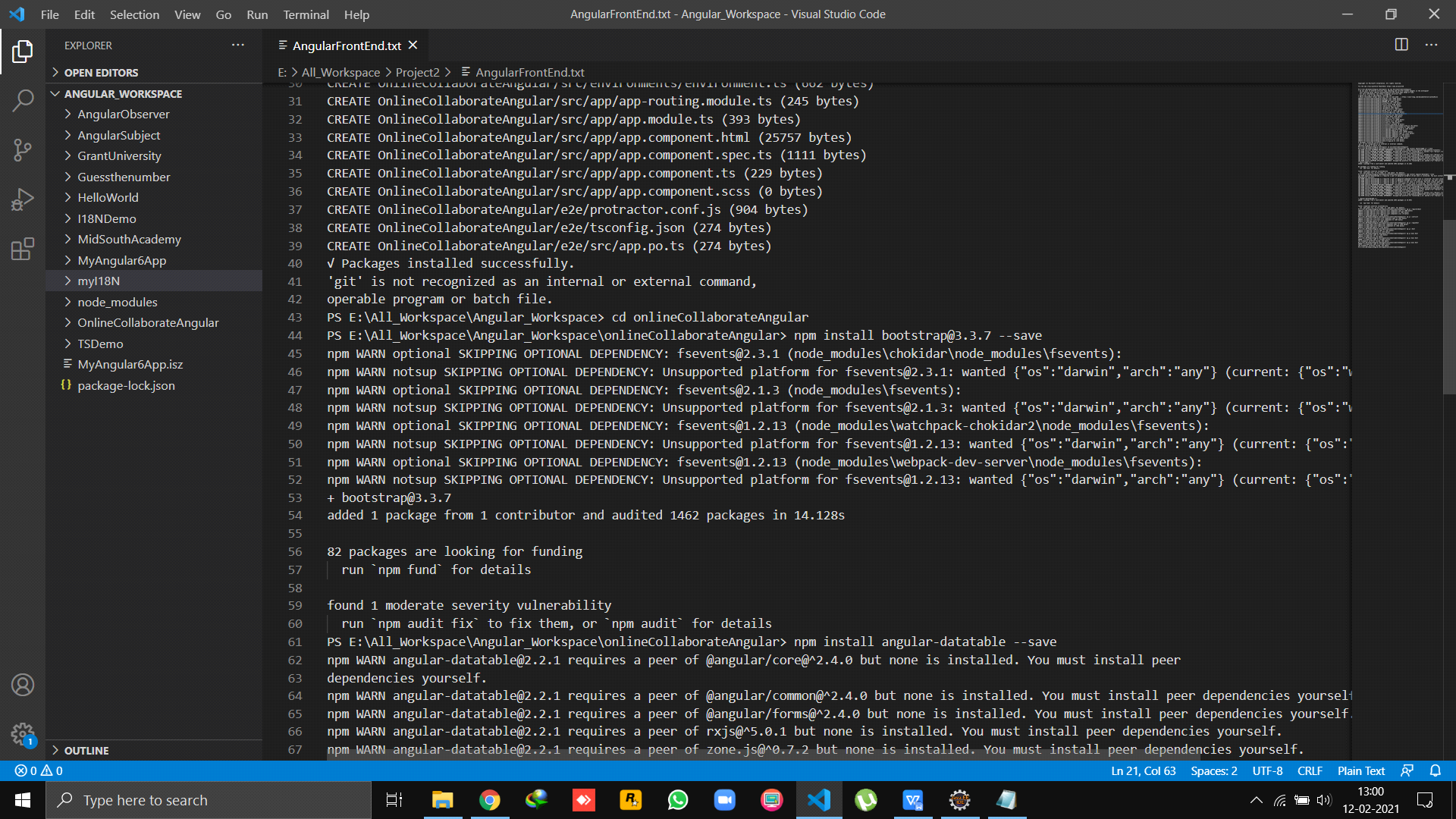
ng new OnlineCollaborateAngular



12.Install Bootstrap CSS framework

Use the following command to install bootstrap in the project.

*E:\All\_Workspace\Angular\_Workspace\OnlineCollaborateAngular>* ***npm install bootstrap@3.3.7 --save***



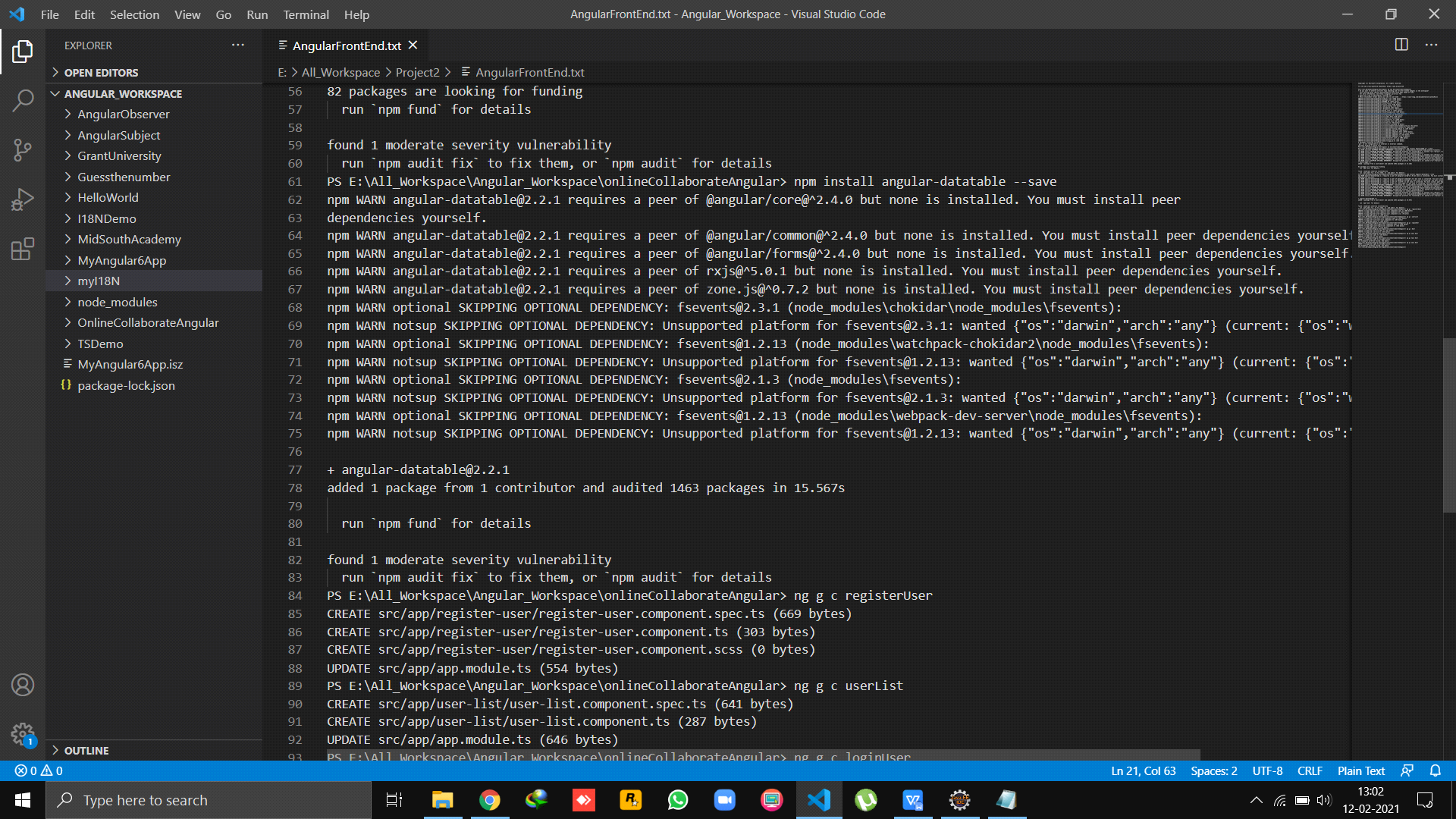
13.Now, include the following code in the style.css file.

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

14.Install Angular Data Table

Use the following command to install bootstrap in the project.

*E:\All\_Workspace\Angular\_Workspace\OnlineCollaborateAngular>* ***npm install angular-datatable --save***



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

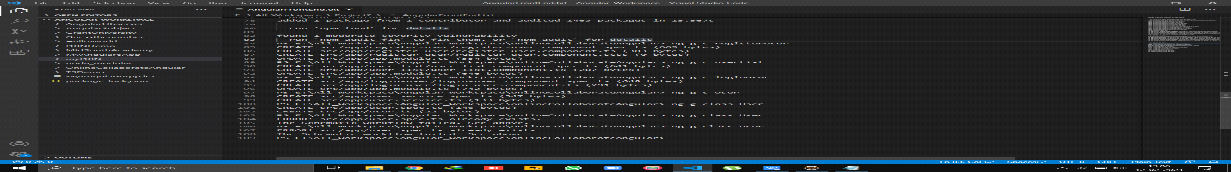
ng g c registerUser

ng g c userList

ng g c loginUser

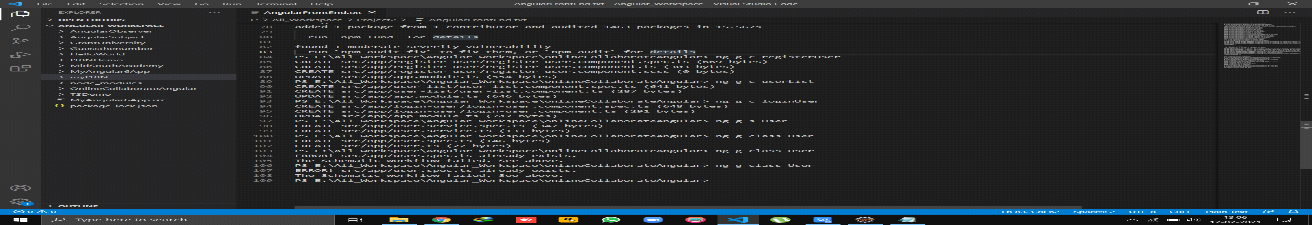






1. Let's also create a service class by using the following command: -

*ng g s User*



1. Edit the **app.module.ts** file

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

**//app.module.ts**

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

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

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

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

import {DataTablesModule } from 'angular-datatables';

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

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

import { RegisterUserComponent } from './register-user/register-user.component';

import { UserListComponent } from './user-list/user-list.component';

import { LoginUserComponent } from './login-user/login-user.component';

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

import { NavigationComponent } from './navigation/navigation.component';

import { LayoutModule } from '@angular/cdk/layout';

import { MatToolbarModule } from '@angular/material/toolbar';

import { MatButtonModule } from '@angular/material/button';

import { MatSidenavModule } from '@angular/material/sidenav';

import { MatIconModule } from '@angular/material/icon';

import { MatListModule } from '@angular/material/list';

import { UserProfileComponent } from './user-profile/user-profile.component';

import { BlogListComponent } from './blog-list/blog-list.component';

import { ActiveUserComponent } from './active-user/active-user.component';

import { HomeComponent } from './home/home.component';

import { NavigationUserComponent } from './navigation-user/navigation-user.component';

import { AboutUsComponent } from './about-us/about-us.component';

import { EditorModule } from '@tinymce/tinymce-angular';

@NgModule({

declarations: [

AppComponent,

RegisterUserComponent,

UserListComponent,

LoginUserComponent,

NavigationComponent,

UserProfileComponent,

BlogListComponent,

ActiveUserComponent,

HomeComponent,

NavigationUserComponent,

AboutUsComponent,

],

imports: [

BrowserModule,

AppRoutingModule,

FormsModule,

ReactiveFormsModule,

HttpClientModule,

EditorModule,

DataTablesModule, BrowserAnimationsModule, LayoutModule, MatToolbarModule, MatButtonModule, MatSidenavModule, MatIconModule, MatListModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

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

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

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

import { RegisterUserComponent } from './register-user/register-user.component';

import { UserListComponent } from './user-list/user-list.component';

import { LoginUserComponent } from './login-user/login-user.component';

import {ActiveUserComponent} from './active-user/active-user.component';

import {BlogListComponent } from './blog-list/blog-list.component';

import {UserProfileComponent} from './user-profile/user-profile.component';

import { HomeComponent } from './home/home.component';

import { NavigationComponent } from './navigation/navigation.component';

import { NavigationUserComponent } from './navigation-user/navigation-user.component';

import { AboutUsComponent } from './about-us/about-us.component';

const routes: Routes = [

{ path: '', redirectTo: 'home', pathMatch: 'full' },

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

{path: 'about-us',component:AboutUsComponent},

{path:'nav/:Id', component:NavigationComponent,

children: [

{ path: 'user-list', component: UserListComponent },

{ path: 'active-user' , component: ActiveUserComponent},

{ path: 'blog-list' , component: BlogListComponent},

{ path: 'user-profile/:Id' , component: UserProfileComponent}

]

} ,

{path:'nav-user/:Id', component:NavigationUserComponent,

children:[

{ path: 'blog-list' , component: BlogListComponent},

{ path: 'user-profile/:Id' , component: UserProfileComponent},

]

},

{ path: 'register-user', component: RegisterUserComponent },

{ path: 'login-user' , component: LoginUserComponent} ,

];

@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule]

})

export class AppRoutingModule { }

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

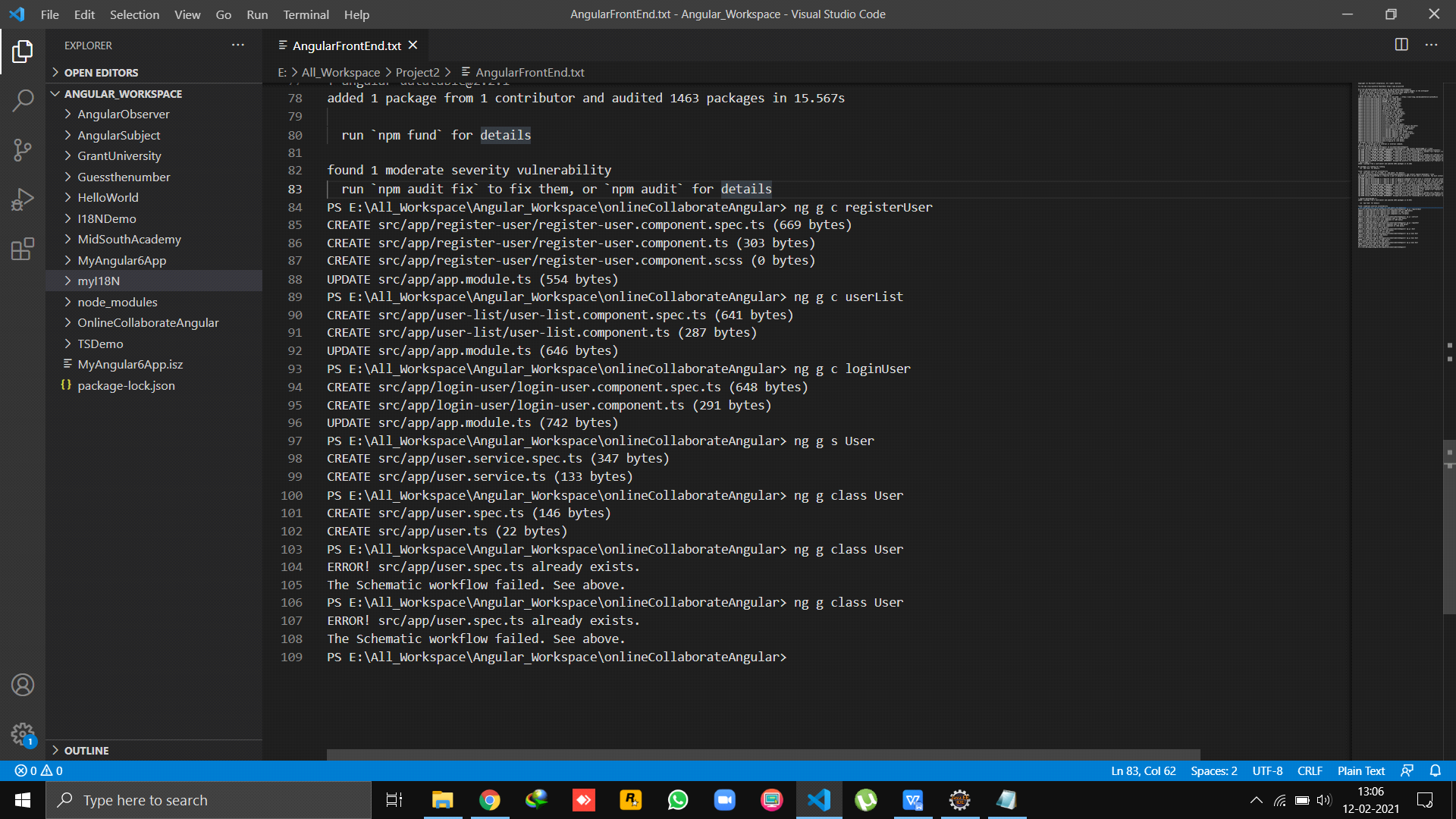
<router-outlet>

</router-outlet>

21 .Create the **User.ts** class

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

*ng g class User*



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

export class User {

user\_id!: number;

first\_name!: String;

last\_name!: String;

username!: String;

password!: String;

confirm\_password!: String;

email!: String;

role!: String;

status!: String;

isOnline!: boolean;

enabled!: boolean;

}

23 . Edit the **User.service.ts**file

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

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

import { Observable } from 'rxjs';

@Injectable({

providedIn: 'root'

})

export class UserService {

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

constructor(private http:HttpClient) {

}

getUserList(): Observable<any> {

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

}

createUser(user: object): Observable<object> {

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

}

deleteUser(userId: number): Observable<any> {

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

}

getUser(userId: number): Observable<Object> {

return this.http.get(`${this.baseUrl}/user/${userId}`);

}

updateUser(userId: number, value: any): Observable<Object> {

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

}

deactiveList():Observable<any> {

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

}

activeUser(userId: number): Observable<Object> {

return this.http.post(`${this.baseUrl}/active-user/${userId}`, {responseType: 'text'});

}

checkUser(user: object): Observable<any> {

return this.http.post(`${this.baseUrl}`+"validate-user", user);

}

logoutUser(userId: number): Observable<Object> {

return this.http.post(`${this.baseUrl}/logout-user/${userId}`, {responseType: 'text'});

}

}

24 .Edit the **register-user.component.ts**file

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

import { UserService } from '../user.service';

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

import { User } from '../user';

@Component({

selector: 'app-register-user',

templateUrl: './register-user.component.html',

styleUrls: ['./register-user.component.scss']

})

export class RegisterUserComponent implements OnInit {

constructor(private userservice:UserService) { }

user : User=new User();

submitted = false;

ngOnInit(): void {

this.submitted=false;

}

registrationform=new FormGroup({

firstName:new FormControl('' , [Validators.required ] ),

lastName:new FormControl('' , [Validators.required ] ),

username:new FormControl('' , [Validators.required ] ),

password:new FormControl('' , [Validators.required ] ),

confirm\_password:new FormControl('' , [Validators.required ] ),

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

role:new FormControl()

});

register()

{

this.user=new User();

this.user.firstName=this.FirstName!.value;

this.user.lastName=this.LastName!.value;

this.user.username=this.Username!.value;

if(this.Password!.value===this.ConfirmPassword!.value)

this.user.password=this.Password!.value;

this.user.email=this.Email!.value;

this.user.role=this.Role!.value;

if(this.user.role==="Admin"){

this.user.enabled="true";

this.user.status="Active";

}

else{

this.user.enabled="false";

this.user.status="Inactive";

}

this.user.isOnline="false";

this.submitted = true;

console.log(this.user.firstName);

this.save();

}

save() {

this.userservice.createUser(this.user)

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

this.user = new User();

}

get FirstName(){

return this.registrationform.get('firstName');

}

get LastName(){

return this.registrationform.get('lastName');

}

get Username(){

return this.registrationform.get('username');

}

get Password(){

return this.registrationform.get('password');

}

get ConfirmPassword(){

return this.registrationform.get('confirm\_password');

}

get Email(){

return this.registrationform.get('email');

}

get Role(){

return this.registrationform.get('role');

}

registrationForm(){

this.submitted=false;

this.registrationform.reset();

}

logInUser() {

$(".pages").css("visibility","visible");

$(".peru").css("visibility","hidden");

}

}

25 . Edit the **register-user.component.html** file

<h3 style="text-align: center">Register User</h3>

<div class="row">

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

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

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

<form [formGroup]="registrationform" (ngSubmit)="register()">

<div class="form-group">

<label for="firstName">First Name</label>

<input type="text" class="form-control" formControlName="firstName" data-toggle="tooltip"

data-placement="right" title="Enter First Name" placeholder="Enter Your FirstName">

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

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

<span \*ngIf="FirstName.errors.required">First Name is Required</span>

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

MinLength Error

</span>

</div>

</div>

<div class="form-group">

<label for="lastName">Last Name</label>

<input type="text" class="form-control" formControlName="lastName" data-toggle="tooltip"

data-placement="right" title="Enter Last Name" placeholder="Enter Your LastName">

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

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

<span \*ngIf="LastName.errors.required">LastName is Required</span>

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

MinLength Error

</span>

</div>

</div>

<div class="form-group">

<label for="username">Username</label>

<input type="text" class="form-control" formControlName="username" data-toggle="tooltip"

data-placement="right" title="Enter Username" placeholder="Enter Your Username">

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

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

<span \*ngIf="Username.errors.required">Username is Required</span>

<span \*ngIf = "Username.errors.minlength"> MinLength Error </span>

</div>

</div>

<div class="form-group">

<label for="password">Password</label>

<input type="password" class="form-control" formControlName="password" data-toggle="tooltip"

data-placement="right" title="Enter Password" placeholder="Enter Your Password">

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

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

<span \*ngIf="Password.errors.required">Password is Required</span>

<span \*ngIf = "Password.errors.minlength"> MinLength Error </span>

</div>

</div>

<div class="form-group">

<label for="confirm\_password">Confirm Password</label>

<input type="password" class="form-control" formControlName="confirm\_password" data-toggle="tooltip"

data-placement="right" title="Enter Confirm Password" placeholder="Enter Your Confirm Password">

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

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

<span \*ngIf="ConfirmPassword.errors.required">Confirm Password is Required</span>

<span \*ngIf="ConfirmPassword.errors.pattern">Password and Confirm Password does not match.</span>

<span \*ngIf = "ConfirmPassword.errors.minlength"> MinLength Error </span>

</div>

</div>

<div class="form-group">

<label for="email">Email</label>

<input type="text" class="form-control" formControlName="email"

data-toggle="tooltip" data-placement="right" title="Enter Email Id" placeholder="Enter Your Email Id">

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

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

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

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

Invalid Email Format

</span>

</div>

</div>

<div class="form-group">

<label for="role">Role</label>

<select class="form-control" formControlName="role" data-toggle="tooltip"

data-placement="right" title="Select Role" >

<option value="null">--User Role--</option>

<option value="Admin">Admin</option>

<option value="User">User</option>

</select>

</div>

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

</form>

</div>

</div>

</div>

<div [hidden]="!submitted">

<h4 style="font-weight: bold; font-style: italic; text-align: center; color:green;">Congratulations..!!! You have Registered successFully..!!</h4>

</div>

Edit the **register-user.component.scss** file

body {

margin: 0;

padding: 0;

font-family: sans-serif;

background: linear-gradient(to right, #b92b27, #1565c0)

}

.row {

width: 300px;

left: 150%;

padding: 20px;

position:relative;

text-align:center;

transition: 0.25s;

margin-top: 125px

}

.row input[type="text"],

.row input[type="password"],

.row select {

border: 0;

background: none;

display: block;

margin: 10px auto;

text-align: center;

border: 3px solid #3498db;

padding: 5px 10px;

width: 200px;

outline: none;

color: white;

border-radius: 25px;

transition: 0.25s

}

.row label{

color:aliceblue;

}

.row h2,h4 {

color: white;

text-transform: uppercase;

font-weight: 500;

}

.row label:hover{

color: yellow;

}

.row h2:hover{

color:yellow;

}

.row a{

color:white;

}

.row a:hover{

color:palegoldenrod;

}

.row input[type="text"]:focus,

.row input[type="password"]:focus,

.row select:focus {

width: 300px;

border-color: #2ecc71

}

.row option{

background-color:black;

}

.row button[type="submit"] {

border: 0;

background: none;

display: block;

margin: 20px auto;

text-align: center;

border: 2px solid #2ecc71;

padding: 14px 40px;

outline: none;

color: white;

border-radius: 24px;

transition: 0.25s;

cursor: pointer

}

.row

button[type="submit"]:hover {

background:rebeccapurple

}

::placeholder {

color:white;

opacity: 1; /\* Firefox \*/

}

:-ms-input-placeholder { /\* Internet Explorer 10-11 \*/

color:white;

}

::-ms-input-placeholder { /\* Microsoft Edge \*/

color:white;

}

.pages{

position:absolute;

top:10%;

left:10%;

visibility: hidden;

}

.form-control{

width:100%;

border-radius:5px;

border: 2px solid rgba(76, 46, 207, 0.856);

font-size: 13px;

font-family:monospace;

}

26 .Edit the**user-list.component.ts**file

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

import { UserService } from '../user.service';

import { User } from '../user';

import { Observable, Subject } from 'rxjs';

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

import { DataTablesModule } from 'angular-datatables';

@Component({

selector: 'app-user-list',

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

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

})

export class UserListComponent implements OnInit {

constructor(private userservice: UserService) { }

userArray: any[] = [];

dtOptions: DataTables.Settings = {};

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

users:any;

user: User = new User();

deleteMessage = false;

userlist: any;

isupdated = false;

ngOnInit() {

this.isupdated = false;

this.dtOptions = {

pageLength: 6,

stateSave: true,

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

processing: true

};

this.userservice.getUserList().subscribe((data) => {

this.users = data;

this.dtTrigger.next();

})

}

deleteUser(userId: number) {

this.userservice.deleteUser(userId)

.subscribe(

(data) => {

console.log(data);

this.deleteMessage = true;

this.userservice.getUserList().subscribe((data) => {

this.users = data

})

},

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

}

updatUser(id: number) {

this.userservice.getUser(id)

.subscribe(

(data) => {

this.userlist = data;

console.log(this.userlist);

}),

(error:any) => console.log(error);

}

userupdateform = new FormGroup({

userId: new FormControl(),

firstName: new FormControl(),

lastName: new FormControl(),

username: new FormControl(),

password: new FormControl(),

email: new FormControl(),

role: new FormControl(),

status: new FormControl(),

isOnline: new FormControl(),

enabled: new FormControl()

});

updateUser(updateusers: any) {

this.user = new User();

this.user.userId = this.UserId!.value;

this.user.firstName = this.FirstName!.value;

this.user.lastName = this.LastName!.value;

this.user.username = this.UserName!.value;

this.user.password = this.Password!.value;

this.user.email = this.Email!.value;

this.user.role = this.Role!.value;

this.user.status = this.Status!.value;

this.user.isOnline = this.IsOnline!.value;

this.user.enabled = this.Enabled!.value;

console.log(this.FirstName!.value);

this.userservice.updateUser(this.user.userId, this.user).subscribe(

data => {

this.isupdated = true;

this.userservice.getUserList().subscribe(data => {

this.users = data

})

},

error => {

console.log(this.users);

console.log(error)});

}

get FirstName() {

return this.userupdateform.get('firstName');

}

get LastName() {

return this.userupdateform.get('lastName');

}

get UserName() {

return this.userupdateform.get('username');

}

get Password() {

return this.userupdateform.get('password');

}

get Email() {

return this.userupdateform.get('email');

}

get Role() {

return this.userupdateform.get('role');

}

get Status() {

return this.userupdateform.get('status');

}

get IsOnline() {

return this.userupdateform.get('isOnline');

}

get Enabled() {

return this.userupdateform.get('enabled');

}

get UserId() {

return this.userupdateform.get('userId');

}

changeisUpdate() {

this.isupdated = false;

}

}

27 . Edit the **user-list.component.html**file

<div class="panel">

<div class="panel panel-default">

<div class="panel-heading">

<h1 style="font-weight: bold; font-style: italic; text-align: center; ">Users</h1><br>

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

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

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

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

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

<strong>User Data Deleted</strong>

</div>

</div>

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

</div>

</div>

<div class="panel-body">

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

<thead class="thead-light">

<tr>

<th>UserID</th>

<th>FirstName</th>

<th>LastName</th>

<th>UserName</th>

<th>Email</th>

<th>Role</th>

<th>Status</th>

<th>IsOnline</th>

<th>Enabled</th>

<th>Action</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let user of users">

<td>{{user.userId}}</td>

<td>{{user.firstName}}</td>

<td>{{user.lastName}}</td>

<td>{{user.username}}</td>

<td>{{user.email}}</td>

<td>{{user.role}}</td>

<td>{{user.status}}</td>

<td>{{user.isOnline}}</td>

<td>{{user.enabled}}</td>

<td><button (click)="deleteUser(user.userId)" class='btn btn-primary'><i class="fa fa-futboll-0">Delete</i></button> &nbsp; &nbsp;

<button (click)="updatUser(user.userId)" class='btn btn-info' data-toggle="modal"

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

</td>

</tr>

</tbody><br>

</table>

</div>

</div>

</div>

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

<div class="modal-dialog">

<div class="modal-content">

<form [formGroup]="userupdateform" #updateusers (ngSubmit)="updateUser(updateusers)">

<!-- Modal Header -->

<div class="modal-header">

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

</div>

<!-- Modal body -->

<div class="modal-body" \*ngIf="userlist">

<div [hidden]="isupdated">

<input type="hidden" class="form-control" formControlName="userId" [(ngModel)]="userlist.userId">

<div class="form-group">

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

<input type="text" class="form-control" formControlName="firstName" [(ngModel)]="userlist.firstName">

</div>

<div class="form-group">

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

<input type="text" class="form-control" formControlName="lastName" [(ngModel)]="userlist.lastName">

</div>

<div class="form-group">

<label for="name">Username</label>

<input type="text" class="form-control" formControlName="username" [(ngModel)]="userlist.username">

</div>

<div class="form-group">

<label for="name">Password</label>

<input type="password" class="form-control" formControlName="password" [(ngModel)]="userlist.password">

</div>

<div class="form-group">

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

<input type="text" class="form-control" formControlName="email" [(ngModel)]="userlist.email">

</div>

<div class="form-group">

<label for="name">Role</label>

<select class="form-control" formControlName="role" required>

<option value="Admin" [selected]="'Admin' == userlist.role">Admin</option>

<option value="User" [selected]="'User' == userlist.role">User</option>

</select>

</div>

<div class="form-group">

<label for="name">Status</label>

<input type="text" class="form-control" formControlName="status" [(ngModel)]="userlist.status">

</div>

<div class="form-group">

<label for="name">IsOnline</label>

<input type="text" class="form-control" formControlName="isOnline" [(ngModel)]="userlist.isOnline">

</div>

<div class="form-group">

<label for="name">Enabled</label>

<input type="text" class="form-control" formControlName="enabled" [(ngModel)]="userlist.enabled">

</div>

</div>

<div [hidden]="!isupdated">

<h4>User Detail Updated!</h4>

</div>

</div>

<!-- Modal footer -->

<div class="modal-footer">

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

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

</div>

</form>

</div>

</div>

</div>

Edit the **user-list.component.scss**file

.panel{

position:relative;

left:95px;

background-color:lightskyblue;

}

td {

border-right: solid 0.5px rgb(10, 10, 10);

border-bottom:solid 0.5px black;

}

th{

border-top:solid 0.5px black;

border-bottom:solid 0.5px black; ;

}

.panel th:hover{

color:white;

}

.panel tr:hover{

font-weight: bold;

color:blue;

}

.panel button:hover{

font-weight: bolder;

color:black;

}

.panel td:nth-child(1n+1):hover{

color:black;

}

28 **. Install following:**

npm install jquery --save

npm install datatables.net --save

npm install datatables.net-dt --save

npm install angular-datatables@6.0.0 --save

npm install @types/jquery --save-dev

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

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

"styles": [

"src/styles.scss",

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

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

],

"scripts": [

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

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

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

]

30.**Save All.**

31.**Run the SpringBoot Application**

32.**Run the Angular project**

**Login -User Component**

**Login-user.component.ts**

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

import { UserService } from '../user.service';

import { User } from '../user';

import { Observable, Subject } from 'rxjs';

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

import { DataTablesModule } from 'angular-datatables';

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

@Component({

selector: 'app-login-user',

templateUrl: './login-user.component.html',

styleUrls: ['./login-user.component.scss']

})

export class LoginUserComponent implements OnInit {

user : User=new User();

currentUser : any;

constructor(private userService: UserService, private router: Router) { }

ngOnInit(): void {

}

loginform=new FormGroup({

username:new FormControl('',[Validators.required]),

password:new FormControl('',[Validators.required])

})

validateUser() {

this.user=new User();

this.user.username=this.Username?.value;

this.user.password=this.Password?.value;

this.userService.checkUser(this.user).subscribe (

data => {

console.log(data);

if(data!=null) {

this.currentUser=data;

if(this.currentUser.role==="Admin"){

this.router.navigateByUrl("/nav/" +`${this.currentUser.userId}`);

}

else{

this.router.navigateByUrl("/nav-user/"+`${this.currentUser.userId}`);

}

}

else {

console.log("Object Empty");

}

},

error => console.log(error)

)

}

get Username() {

return this.loginform.get ('username');

}

get Password() {

return this.loginform.get ('password');

}

RegUser(){

$(".pagess").css("visibility","visible");

$(".card").css("visibility","hidden");

}

}

**Login-user.component.html**

<div class="login-box">

<h2>Login</h2>

<form [formGroup]="loginform" (ngSubmit)="validateUser()">

<!--Username-->

<div class="user-box">

<input type="text" class="form-control" formControlName="username" data-toggle="tooltip"

data-placement="right" title="Enter Username" placeholder="Username">

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

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

<span \*ngIf="Username.errors.required">Username is Required</span>

</div>

</div>

<!--Password-->

<div class="user-box">

<input type="password" class="form-control" formControlName="password" data-toggle="tooltip"

data-placement="right" title="Enter Password" placeholder="Password">

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

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

<span \*ngIf="Password.errors.required">Password is Required</span>

</div>

</div>

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

</form>

</div>

**Login-user.component.scss**

html {

height: 100%;

}

body {

margin:0;

padding:0;

font-family: sans-serif;

background: linear-gradient(#141e30, #243b55);

}

.login-box {

position:fixed;

top: 50%;

right: 50%;

width: 400px;

padding: 40px;

transform: translate(-50%, -50%);

background: rgba(6, 1, 31, 0.938);

box-sizing: border-box;

box-shadow: 0 15px 25px rgba(0,0,0,.6);

border-radius: 20px;

}

.login-box h2 {

margin: 0 0 30px;

padding: 0;

color: #fff;

text-align: center;

}

.login-box .user-box {

position: relative;

}

.login-box .user-box input {

width: 100%;

padding: 10px 0;

font-size: 16px;

color: #fff;

margin-bottom: 30px;

border: none;

border-bottom: 1px solid #fff;

outline: none;

background: transparent;

}

.login-box .user-box label {

position: absolute;

top:0;

left: 0;

padding: 10px 0;

font-size: 16px;

color: #fff;

pointer-events: none;

transition: .5s;

}

.login-box .user-box input:focus ~ label,

.login-box .user-box input:valid ~ label {

top: -20px;

left: 0;

color: #03e9f4;

font-size: 12px;

}

.login-box form a {

position: relative;

display: inline-block;

padding: 10px 20px;

color: #03e9f4;

font-size: 16px;

text-decoration: none;

text-transform: uppercase;

overflow: hidden;

transition: .5s;

margin-top: 40px;

letter-spacing: 4px

}

.login-box a:hover {

background: #03e9f4;

color: #fff;

border-radius: 5px;

box-shadow: 0 0 5px #03e9f4,

0 0 25px #03e9f4,

0 0 50px #03e9f4,

0 0 100px #03e9f4;

}

.login-box a span {

position: absolute;

display: block;

}

.login-box a span:nth-child(1) {

top: 0;

left: -100%;

width: 100%;

height: 2px;

background: linear-gradient(90deg, transparent, #03e9f4);

animation: btn-anim1 1s linear infinite;

}

@keyframes btn-anim1 {

0% {

left: -100%;

}

50%,100% {

left: 100%;

}

}

.login-box a span:nth-child(2) {

top: -100%;

right: 0;

width: 2px;

height: 100%;

background: linear-gradient(180deg, transparent, #03e9f4);

animation: btn-anim2 1s linear infinite;

animation-delay: .25s

}

@keyframes btn-anim2 {

0% {

top: -100%;

}

50%,100% {

top: 100%;

}

}

.login-box a span:nth-child(3) {

bottom: 0;

right: -100%;

width: 100%;

height: 2px;

background: linear-gradient(270deg, transparent, #03e9f4);

animation: btn-anim3 1s linear infinite;

animation-delay: .5s

}

@keyframes btn-anim3 {

0% {

right: -100%;

}

50%,100% {

right: 100%;

}

}

.login-box a span:nth-child(4) {

bottom: -100%;

left: 0;

width: 2px;

height: 100%;

background: linear-gradient(360deg, transparent, #03e9f4);

animation: btn-anim4 1s linear infinite;

animation-delay: .75s

}

@keyframes btn-anim4 {

0% {

bottom: -100%;

}

50%,100% {

bottom: 100%;

}

}

**Active User Component**

**Active-user.component.ts**

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

import { UserService } from '../user.service';

import { User } from '../user';

import { Observable, Subject } from 'rxjs';

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

import { DataTablesModule } from 'angular-datatables';

@Component({

selector: 'app-active-user',

templateUrl: './active-user.component.html',

styleUrls: ['./active-user.component.scss']

})

export class ActiveUserComponent implements OnInit {

dtOptions: DataTables.Settings={};

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

constructor(private userservice: UserService) { }

users:any;

user: User =new User();

deactiveList: any;

isEnabled= false;

ngOnInit(){

this.isEnabled= false;

this.dtOptions={

pageLength: 6,

stateSave: true,

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

processing: true

};

this.userservice.deactiveList().subscribe(data =>{

this.users =data;

this.dtTrigger.next();

})

}

enabledUser(id: number){

this.userservice.activeUser(id).

subscribe(

(data) =>{

this.deactiveList =data;

console.log(this.deactiveList);

this.activeUser(id);

},

(error) => console.log(error)

);

}

activeUserForm = new FormGroup({

userId: new FormControl()

});

activeUser(id:number){

this.user =new User();

this.user.userId=id;

this.userservice.activeUser(this.user.userId).subscribe(

data => {

this.userservice.deactiveList().subscribe( data =>{

this.users =data

console.log(this.users)

} )

},

error => console.log(error));

}

get UserId() {

return this.activeUserForm.get('userId');

}

}

**Active-user.component.html**

<h1 style="font-weight: bold; font-style: italic; text-align: center; ">Users Deactive List</h1><br>

<div class="panel">

<div class="panel panel-default">

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

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

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

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

<h2 style="text-align: center; font-weight: bold;">User Activated</h2>

</div>

</div>

</div>

<div class="panel-body">

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

<thead class="thead-light">

<tr>

<th>UserID</th>

<th>Name</th>

<th>UserName</th>

<th>Enabled</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let user of users">

<td>{{user.userId}}</td>

<td>{{user.firstName }}</td>

<td>{{user.username}}</td>

<td>{{user.enabled}}</td>

<td> <button (click)="enabledUser(user.userId)" class='btn btn-info' data-toggle="modal"

data-target="#myModal">Activate</button> &nbsp; &nbsp;</td>

</tr>

</tbody>

</table>

</div>

</div>

</div>

**Active-user.component.scss**

.panel{

position:relative;

left:95px;

background-color:lightskyblue;

}

td {

border-right: solid 0.3px rgb(10, 10, 10);

border-bottom:solid 0.3px black;

}

th{

border-top:solid 0.5px black;

border-bottom:solid 0.5px black; ;

}

.panel-body th:hover{

color:white;

}

.panel-body tr:hover{

font-weight: bold;

color:blue;

}

.panel button:hover{

font-weight: bolder;

color:black;

}

.panel-body td:nth-child(1n+1):hover{

color:black;

}

**User-profile component**

**User-profile.component.ts**

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

import { UserService } from '../user.service';

import { User } from '../user';

import { Observable, Subject } from 'rxjs';

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

import { DataTablesModule } from 'angular-datatables';

import { param } from 'jquery';

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

@Component({

selector: 'app-user-profile',

templateUrl: './user-profile.component.html',

styleUrls: ['./user-profile.component.scss']

})

export class UserProfileComponent implements OnInit {

Id?:number;

user:any;

constructor( private route: ActivatedRoute, private userservice: UserService){ }

ngOnInit(){

this.route.params.subscribe (

(params:Params)=> {

this.Id=+params["Id"];

console.log(this.Id);

this.userservice.getUser(this.Id).subscribe (

data=>{

this.user=data;

console.log(this.user);

}

)

}

)

}

}

**User-profile.component.html**

<div>

<div class="main">

<div class="page-content page-container" id="page-content">

<div class="padding">

<div class="row container d-flex justify-content-center">

<div class="col-xl-6 col-md-12">

<div class="card user-card-full">

<div class="row m-l-0 m-r-0">

<div class="col-sm-4 bg-c-lite-green user-profile">

<div class="card-block text-center text-white">

<div class="m-b-25"> <img src="https://img.icons8.com/bubbles/100/000000/user.png" class="img-radius" alt="User-Profile-Image"> </div>

<h6 class="f-w-600">{{user.username}}</h6>

<p>{{user.role}}</p> <i class=" mdi mdi-square-edit-outline feather icon-edit m-t-10 f-16"></i>

</div>

</div>

<div class="col-sm-8">

<div class="card-block">

<h6 class="m-b-20 p-b-5 b-b-default f-w-600">Information</h6>

<div class="row">

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

<p class="m-b-10 f-w-600">Name</p>

<h6 class="text-muted f-w-400">{{user.firstName +" "+ user.lastName}}</h6>

</div>

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

<p class="m-b-10 f-w-600">Email</p>

<h6 class="text-muted f-w-400">{{user.email}}</h6>

</div>

</div>

</div>

</div>

</div>

</div> </div </div> </div>

**User-profile.component.scss**

.main{

position:relative;

left:95px;

}

body {

background-color: #f9f9fa;

position: static;

left:100px;

}

.padding {

padding: 3rem !important

}

.user-card-full {

overflow: hidden

}

h6:hover{

color:blue;

font-weight: bold;

}

p:hover{

color:blue;

font-weight: bold;

}

.card {

border-radius: 5px;

-webkit-box-shadow: 0 1px 20px 0 rgba(69, 90, 100, 0.08);

box-shadow: 0 1px 20px 0 rgba(69, 90, 100, 0.08);

border: none;

margin-bottom: 30px

}

.m-r-0 {

margin-right: 0px

}

.m-l-0 {

margin-left: 0px

}

.user-card-full .user-profile {

border-radius: 5px 0 0 5px

}

.bg-c-lite-green {

background: -webkit-gradient(linear, left top, right top, from(#f29263), to(#ee5a6f));

background: linear-gradient(to right, #ee5a6f, #f29263)

}

.user-profile {

padding: 20px 0

}

.card-block {

padding: 1.25rem

}

.m-b-25 {

margin-bottom: 25px

}

.img-radius {

border-radius: 5px

}

h6 {

font-size: 14px

}

.card .card-block p {

line-height: 25px

}

@media only screen and (min-width: 1400px) {

p {

font-size: 14px

}

}

.card-block {

padding: 1.25rem

}

.b-b-default {

border-bottom: 1px solid #e0e0e0

}

.m-b-20 {

margin-bottom: 20px

}

.p-b-5 {

padding-bottom: 5px !important

}

.card .card-block p {

line-height: 25px

}

.m-b-10 {

margin-bottom: 10px

}

.text-muted {

color: #919aa3 !important

}

.b-b-default {

border-bottom: 1px solid #e0e0e0

}

.f-w-600 {

font-weight: 600

}

.m-b-20 {

margin-bottom: 20px

}

.m-t-40 {

margin-top: 20px

}

.p-b-5 {

padding-bottom: 5px !important

}

.m-b-10 {

margin-bottom: 10px

}

.m-t-40 {

margin-top: 20px

}

.user-card-full .social-link li {

display: inline-block

}

.user-card-full .social-link li a {

font-size: 20px;

margin: 0 10px 0 0;

-webkit-transition: all 0.3s ease-in-out;

transition: all 0.3s ease-in-out

}

**About-us component**

**About-us.component.ts**

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

@Component({

selector: 'app-about-us',

templateUrl: './about-us.component.html',

styleUrls: ['./about-us.component.scss']

})

export class AboutUsComponent implements OnInit {

constructor() { }

ngOnInit(): void {

}

}

**About-us.component.html**

<h1 class="logo"><span>Relaxato </span>City</h1>

<html>

<head>

<style>

p {

color: navy;

text-indent: 30px;

text-transform: uppercase;

}

</style>

</head>

<body>

<p style="text-align:center">

Welcome to Relaxato !!!

<br>

Relaxato City is an Photography Blogging Website.

<br>

Here you can share Photos,vlogs

<br>

You can also share your travel experiences with others

</p>

</body>

</html>

About-us.component.scss

.logo{

font-size: 2.5rem; font-weight: 600; text-transform: uppercase;

letter-spacing: 10px; line-height: 4rem; margin-top: 20px;

color:crimson;

}

.logo span{ font-size: 2.5rem;margin-left: 40pc; margin-right: 50px; color:chocolate; }

**Blog Component**

**Blog.ts**

export class Blog{

blogId!:number;

blogTitel!: String;

blogContent!: String;

blogPosted:any;

status!: String;

noOfLikes!: number;

noFoComments!: number;

noOfViews!: number;

userId!: number;

username!: String;

}

**Blog.service.ts**

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

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

import { Observable } from 'rxjs';

@Injectable({

providedIn: 'root'

})

export class BlogService {

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

constructor(private http:HttpClient){ }

getBlogList(): Observable<any>{

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

}

createBlog(blog: object): Observable<object>{

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

}

deletBlog(blogId:number): Observable<any>{

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

}

getBlog(blogId:number): Observable<Object>{

return this.http.get(`${this.baseUrl}/blog/${blogId}`);

}

updateBlog(blogId:number, value:any): Observable<Object>{

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

}

}

**Blog-list.component.ts**

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

import {BlogService} from '../blog.service';

import { Blog } from '../blog';

import { Observable, Subject } from 'rxjs';

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

import { DataTablesModule } from 'angular-datatables';

@Component({

selector: 'app-blog-list',

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

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

})

export class BlogListComponent implements OnInit {

constructor(private blogservice: BlogService) { }

blog: Blog =new Blog();

submitted = false;

ngOnInit(): void {

this.submitted = false;

}

}

**Blog-list.component.html**

<div class="main">

<div class="container mt-5">

<div class="d-flex justify-content-center row">

<div class="col-md-8">

<div class="d-flex flex-column comment-section">

<div class="bg-white p-2">

<div class="d-flex flex-row user-info">

<div class="d-flex flex-column justify-content-start ml-2"><span class="d-block font-weight-bold name"><strong></strong></span></div>

</div>

<div class="mt-2">

<p class="comment-text"></p>

</div>

</div>

<div class="bg-light p-2">

<div class="d-flex flex-row align-items-start"><textarea class="form-control ml-1 shadow-none textarea"></textarea></div>

<div class="mt-2 text-right"><button class="btn btn-primary btn-sm shadow-none" type="button">Post comment</button><button class="btn btn-outline-primary btn-sm ml-1 shadow-none" type="button">Cancel</button></div>

</div>

</div>

</div>

</div>

</div>

</div>

**Blog-list.component.scss**

body {

background: #eee

}

.main{

position:relative;

left:95px;

background-color:lightskyblue;

}

.date {

font-size: 11px

}

.comment-text {

font-size: 12px

}

.fs-12 {

font-size: 12px

}

.shadow-none {

box-shadow: none

}

.name {

color: #007bff

}

.cursor:hover {

color: blue

}

.cursor {

cursor: pointer

}

.textarea {

resize: none

}

**Home Component**

**Home.component.ts**

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

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

@Component({

selector: 'app-home',

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

styleUrls: ['./home.component.scss']

})

export class HomeComponent implements OnInit {

constructor(private router : Router) { }

ngOnInit(): void {

}

logInUser() {

$(".pages").css("visibility","visible");

}

RegUser(){

$(".pagess").css("visibility","visible");

}

}

**Home.component.html**

<div class="container">

<div class="navbar">

<div class="menu">

<h1 class="logo"><span>Relaxato </span>City</h1>

<div class="bts">

<a href="/about-us" class="about-us">About us</a>

</div>

</div>

</div>

<div class="main-container">

<div class="main">

<header>

<div class="overlay" id="one">

<h2 class="title">Welcome to Relaxato City</h2>

<div class="bts">

<a mat-list-item class="nav-link" (click)="logInUser()" class="btn">Sign In</a>/<a mat-list-item class="nav-link" (click)="RegUser()" class="btn">Sign Up</a>

</div>

</div>

</header>

</div>

<div class="shadow one"></div>

<div class="shadow two"></div>

</div>

</div>

<div class="pages">

<app-login-user></app-login-user>

</div>

<div class="pagess">

<app-register-user></app-register-user>

</div>

**Home.component.scss**

body{ font-family: 'Recursive', sans-serif; overflow: hidden; }

.container{ max-height: 100vh;

width: 100%; background-color: #131414;

background-image: linear-gradient(135deg, #131414 0%, #000000 100%);

transform-style: preserve-3d; overflow: hidden;

}

.navbar{ position: fixed;

top: 0;

left: 0;

width: 100%;

z-index: 10;

height: 5rem;

}

.menu{ max-width: 72rem;

width: 100%;

height: 100%;

margin: 0 auto;

padding: 0 2rem;

display: flex;

justify-content: space-between;

align-items: center;

color: #fff;

}

.logo{

font-size: 2.5rem; font-weight: 600; text-transform: uppercase;

letter-spacing: 10px; line-height: 4rem; margin-top: 20px;

color:darkgoldenrod;

}

.logo span{ font-size: 2.5rem;margin-left: 10pc; margin-right: 50px; color:darkgoldenrod; }

.hamburger-menu{ height: 4rem; width: 3rem; cursor: pointer;

display: flex; align-items: center; justify-content: flex-end; }

.bar{ width: 1.9rem; height: 1.5px; border-radius: 2px;

background-color: #eee; transition: 0.5s; position: relative; }

.bar::before, .bar::after{ content: ""; position: absolute;

width: inherit; height: inherit; background-color: #eee;

transition: 0.5s; }

.bar::before{ transform: translateY(-9px); }

.bar::after{ transform: translateY(9px); }

.main-container{ overflow: hidden; }

.main{ position: relative; width: 100%; left: 0; z-index: 5;

overflow: hidden; transform-origin: left;

transform-style: preserve-3d; transition: 0.5s;

}

header{ min-height: 100vh; width: 100%;

background: url("image25.jpg") no-repeat top center / cover;

position: relative;

}

.overlay{ position: absolute; width: 100%; height: 100%; top: 0;

left: 0; background-color: rgba(0, 0, 0, 0.712);

display: flex; justify-content: center; align-items: center;

flex-direction: column; color: #fff;

}

.inner{ max-width: 35rem; text-align: center; color: #fff;

padding: 0 2rem;

}

.title{ font-size: 3rem; }

.description{ margin: 10px 0; text-align: center; width: 50%;

font-size: 1.5rem;

}

#one h2:hover{

color:chartreuse;

}

#one .bts{

display:inline-block ;

padding: 0.6rem 0.8rem;

color:crimson;

background-color: transparent;

border:2px solid crimson;

font-size: 1.3rem;

border-radius: 25px;

transition: 3s ease;

transition-property: background-color, color;

}

#one .description:hover{

color:yellow;

}

#one .bts:hover{

color:white;

background-color: crimson;

}

.btn{

color: #fff; text-transform: uppercase;

}

#one .btn:hover{

color:yellow;

}

.container.active .bar{ transform: rotate(360deg);

background-color: transparent;

}

.container.active .bar::before{

transform: translateY(0) rotate(45deg);

}

.container.active .bar::after{

transform: translateY(0) rotate(-45deg);

}

.container.active .main{

animation: main-animation 0.5s ease; cursor: pointer;

transform: perspective(1300px) rotateY(20deg) translateY(10px)

translateZ(310px) scale(0.5);

}

@keyframes main-animation{

from{ transform: translate(0); }

to{

transform: perspective(1300px) rotateY(20deg) translateY(10px)

translateZ(310px) scale(0.5);

}

}

.links{ position: absolute; width: 30%; right: 0; top: 0;

height: 100vh; z-index: 2; overflow: hidden;

display: flex; justify-content: flex-start; align-items: center;

margin-left: 10px;

}

ul{ list-style: none; }

ul li.active a{ color: #e20f2f; }

.links a{ text-decoration: none; color: #eee; padding: 0.7rem 0;

display: inline-block; font-size: 1.8rem; font-weight: 300;

text-transform: uppercase; letter-spacing: 1px; transition: 0.3s;

opacity: 0; transform: translateY(10px);

animation: hide 0.5s forwards ease;

}

.links a:hover{ color: #e20f2f; }

.container.active .links a{

animation: appear 0.5s forwards ease var(--i);

}

.pages{

position:absolute;

top:10%;

left:30%;

visibility: hidden;

}

.pagess{

position:absolute;

top:-20%;

left:35%;

visibility: hidden;

}

**Navigation Components**

**Navigation.component.ts**

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

import { BreakpointObserver, Breakpoints } from '@angular/cdk/layout';

import { Observable } from 'rxjs';

import { map, shareReplay } from 'rxjs/operators';

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

import { UserService } from '../user.service';

import { param } from 'jquery';

@Component({

selector: 'app-navigation',

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

styleUrls: ['./navigation.component.scss']

})

export class NavigationComponent implements OnInit {

Id!: number;

user:any;

isHandset$: Observable<boolean> = this.breakpointObserver.observe(Breakpoints.Handset)

.pipe(

map(result => result.matches),

shareReplay()

);

constructor(private breakpointObserver: BreakpointObserver, private route: ActivatedRoute, private userservice: UserService, private router:Router) {}

ngOnInit(){

this.route.params.subscribe (

(params:Params)=> {

this.Id=+params["Id"];

console.log(this.Id);

this.userservice.getUser(this.Id).subscribe (

data=>{

this.user=data;

console.log(this.user);

}

)

}

)

}

logout(){

this.router.navigateByUrl("/home");

this.userservice.logoutUser(this.Id).subscribe(

( data: any)=>console.log(data)

);

}

}

**Navigation.component.html**

<mat-sidenav-container class="sidenav-container">

<mat-sidenav #drawer class="sidenav" fixedInViewport

[attr.role]="(isHandset$ | async) ? 'dialog' : 'navigation'"

[mode]="(isHandset$ | async) ? 'over' : 'side'"

[opened]="(isHandset$ | async) === false">

<mat-toolbar>Menu</mat-toolbar>

<mat-nav-list>

<a mat-list-item routerLink="user-profile/{{user.userId}}" class="nav-link" class="btn-primary active" role="button">Profile</a>

<a mat-list-item routerLink="blog-list" class="nav-link" class="btn-primary active" role="button">Blog</a>

<a mat-list-item routerLink="user-list" class="nav-link" class="btn-primary active" role="button">View User</a>

<a mat-list-item routerLink="active-user" class="nav-link" class="btn-primary active" role="button">Active User</a>

</mat-nav-list>

</mat-sidenav>

<mat-sidenav-content>

<mat-toolbar color="primary">

<span style="font-style: italic;">RelaxatoCity</span>

<button class="logout" (click)="logout()">LogOut</button>

</mat-toolbar>

</mat-sidenav-content>

</mat-sidenav-container>

<p class="welcome">Welcome &nbsp;<span>{{user.firstName+" "+user.lastName}}</span></p>

<router-outlet></router-outlet>

**Navigation.component.scss**

.sidenav-container {

height: 10%;

width: 1560px;

}

.sidenav {

width: 200px;

}

.sidenav .mat-toolbar {

background: inherit;

}

.mat-toolbar.mat-primary {

position: sticky;

top: 0;

z-index: 1;

}

.logout{

position: absolute;

right:250px;

color:blue;

font-size: small;

border-radius: 8px;

border-color:black;

}

.welcome{

position:relative;

font-style: oblique;

font-weight:large;

left:250px;

top:10px;

}

.welcome span{

color:crimson;

font-family: Verdana, Geneva, Tahoma, sans-serif;

font-size: large;

font-style: italic;

font-weight: bold;

text-transform: uppercase;

}

**Navigation-user.component.ts**

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

import { BreakpointObserver, Breakpoints } from '@angular/cdk/layout';

import { Observable } from 'rxjs';

import { map, shareReplay } from 'rxjs/operators';

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

import { UserService } from '../user.service';

import { param } from 'jquery';

@Component({

selector: 'app-nav-user',

templateUrl: './navigation-user.component.html',

styleUrls: ['./navigation-user.component.scss']

})

export class NavigationUserComponent implements OnInit {

Id?:number;

user:any;

isHandset$: Observable<boolean> = this.breakpointObserver.observe(Breakpoints.Handset)

.pipe(

map(result => result.matches),

shareReplay()

);

constructor(private breakpointObserver: BreakpointObserver, private route: ActivatedRoute, private userservice: UserService, private router:Router) {}

ngOnInit(){

this.route.params.subscribe (

(params:Params)=> {

this.Id=+params["Id"];

console.log(this.Id);

this.userservice.getUser(this.Id).subscribe (

data=>{

this.user=data;

console.log(this.user);

}

)

}

)

}

logout(){

this.router.navigateByUrl("/home");

this.userservice.logoutUser(this.Id).subscribe(

( data: any)=>console.log(data)

);

}

}

**Navigation-user.component.html**

<mat-sidenav-container class="sidenav-container">

<mat-sidenav #drawer class="sidenav" fixedInViewport

[attr.role]="(isHandset$ | async) ? 'dialog' : 'navigation'"

[mode]="(isHandset$ | async) ? 'over' : 'side'"

[opened]="(isHandset$ | async) === false">

<mat-toolbar>Menu</mat-toolbar>

<mat-nav-list>

<a mat-list-item routerLink="user-profile/{{user.userId}}" class="nav-link" class="btn-primary active" role="button">Profile</a>

<a mat-list-item routerLink="blog-list" class="nav-link" class="btn-primary active" role="button">Blog</a>

</mat-nav-list>

</mat-sidenav>

<mat-sidenav-content>

<mat-toolbar color="primary">

<span>RelaxatoCity</span>

<button class="logout" (click)="logout()">LogOut</button>

</mat-toolbar>

</mat-sidenav-content>

</mat-sidenav-container>

<p class="welcome">Welcome &nbsp;<span>{{user.firstName+" "+user.lastName}}</span></p>

<router-outlet></router-outlet>

**Navigation-user.component.scss**

.sidenav-container {

height: 10%;

width: 1560px;

}

.sidenav {

width: 200px;

}

.sidenav .mat-toolbar {

background: inherit;

}

.mat-toolbar.mat-primary {

position: sticky;

top: 0;

z-index: 1;

}

.welcome{

position:relative;

font-style: oblique;

font-weight:large;

left:250px;

top:10px;

}

.welcome span{

color:crimson;

font-family: Verdana, Geneva, Tahoma, sans-serif;

font-size: large;

font-style: italic;

font-weight: bold;

text-transform: uppercase;

}

.logout{

position: absolute;

right:250px;

color:blue;

font-size: small;

border-radius: 8px;

border-color:black;

}

***Personal Observation***

* Spending enough time interacting to get the needed data as much as possible
* Reducing the problem of reactivity. Gaining intuitive understanding of the meaning of structure.
* Addressing problems that are received while compiling the code snippet and sort out the possible solutions.
* Moreover, I personally enjoyed while working in this project and gained more knowledge, boosts my confidence and brings me to successful submit the project on time.