MySQL download-install-setup:

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

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

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

Project:

1. Create a SpringBoot project named 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"*>

<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;

}

}

### OnlineCollaborateApplication

package com.coll.OnlineCollaborate;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class OnlineCollaborateApplication {

public static void main(String[] args) {

SpringApplication.run(OnlineCollaborateApplication.class, args);

}

}

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

### 4(a) 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;

}

}

5.Create the DAO interface implementation class

### 5(a) User.dao

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);

List<User> getAllDeactiveUser();

boolean updateUserProfile(String file, Integer userId);

}

6.Create the DAO interface implementation class

### 6(a) User.daoImpl

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 wheere 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+"'";

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

try {

user=(User)query.getSingleResult();

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();

}

}

7.Create the service layer interface

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

### 7(a) User.service

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);

List<User> getAllDeactiveUser();

boolean updateUserProfile(String file, Integer userId);

}

8.Create the service layer implementation class

### 8(a) User.serviceImpl

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();

}

}

9.Create the controller class

### 9(a) User.controller

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);

}

}

### 5(b) Blog.dao

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);

}

### 6(b) Blog.daoImpl

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;

@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 blogId) {

// TODO Auto-generated method stub

return null;

}

@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(Blog blogId) {

try

{

sessionFactory.getCurrentSession().delete(blogId);

return true;

}

catch(Exception ex)

{

ex.printStackTrace();

return false;

}

}

@Override

public List<Blog> mainList() {

// TODO Auto-generated method stub

return null;

}}

### 7(b) Blog.service

package com.coll.OnlineCollaborate.service;

import java.util.List;

import com.coll.OnlineCollaborate.model.Blog;

public interface IBlogService {

List<Blog> getAllBlogs();

List<Blog> getBlogsByStatus(String status);

List<Blog> getUsersBlogs(int blogId);

List<Blog> mainList();

Blog getBlogById(int blogId);

boolean addBlog(Blog blog);

boolean updateBlog(Blog blog);

boolean deleteBlog(Blog blogId);

}

### 8(b) Blog.serviceImpl

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 List<Blog> mainList() {

return blogDao.mainList();

}

@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(Blog blogId) {

return blogDao.deleteBlog(blogId);

}

}

### 9(b) Blog.controller

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.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/{blog\_id}")

public boolean deleteBlog(@PathVariable("blog\_id") Blog blog\_id)

{

return blogService.deleteBlog(blog\_id);

}

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

public Blog blogById(@PathVariable("blog\_id") int blog\_id)

{

return blogService.getBlogById(blog\_id);

}

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

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

{

return blogService.updateBlog(blog);

}

}

### 4(c) Blogcomments.model

package com.coll.OnlineCollaborate.model;

import java.io.Serializable;

import java.time.LocalDate;

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 title;

int noOfLikes;

String blogComment;

LocalDate currentDate;

@ManyToOne

@JoinColumn(name="BlogId")

@JsonBackReference

Blog blog;

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 getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public int getNoOfLikes() {

return noOfLikes;

}

public void setNoOfLikes(int noOfLikes) {

this.noOfLikes = noOfLikes;

}

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;

}

public static long getSerialversionuid() {

return serialVersionUID;

}

}

### 5(c) Blogcomments.dao

package com.coll.OnlineCollaborate.dao;

import java.util.List;

import com.coll.OnlineCollaborate.model.BlogComments;

public interface IBlogCommentsDao {

List<BlogComments> getAllBlogComments();

BlogComments getBlogCommentsById(int blogCommentId);

boolean addBlogComments(BlogComments blogComments);

boolean updateBlogComments(BlogComments blogComments);

boolean deleteBlogComments(BlogComments blogCommentId);

}

### 6(c) Blogcomments.daoImpl

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.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 blogCommentId) {

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

}

@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(BlogComments blogCommentId) {

try

{

sessionFactory.getCurrentSession().delete(blogCommentId);

return true;

}

catch(Exception ex)

{

ex.printStackTrace();

return false;

}

}

}

### 7(c) Blogcomments.service

package com.coll.OnlineCollaborate.service;

import java.util.List;

import com.coll.OnlineCollaborate.model.BlogComments;

public interface IBlogCommentsService {

List<BlogComments> getAllBlogComments();

BlogComments getBlogCommentsById(int blogCommentId);

boolean addBlogComments(BlogComments blogComments);

boolean updateBlogComments(BlogComments blogComments);

boolean deleteBlogComments(BlogComments blogCommentId);

}

### 8(c) Blogcomments.serviceImpl

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 blogCommentId) {

return blogCommentsDao.getBlogCommentsById(blogCommentId);

}

@Override

public boolean addBlogComments(BlogComments blogComments) {

return blogCommentsDao.addBlogComments(blogComments);

}

@Override

public boolean updateBlogComments(BlogComments blogComments) {

return blogCommentsDao.updateBlogComments(blogComments);

}

@Override

public boolean deleteBlogComments(BlogComments blogCommentId) {

return blogCommentsDao.deleteBlogComments(blogCommentId);

}

}

### 9(c) Blogcomments.controller

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.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 saveBlogComments(@RequestBody BlogComments blogComments)

{

return blogCommentsService.addBlogComments(blogComments);

}

@GetMapping("blogComments-list")

public List<BlogComments> allBlogComments()

{

return blogCommentsService.getAllBlogComments();

}

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

public boolean deleteBlogComments(@PathVariable("blogComment\_id") BlogComments blogComment\_id)

{

return blogCommentsService.deleteBlogComments(blogComment\_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 updateBlog(@PathVariable("blogComments") BlogComments blogComments)

{

return blogCommentsService.updateBlogComments(blogComments);

}

}

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

**# Database**

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";

**# Hibernate**

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

Properties hibernateProperties=new Properties();

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

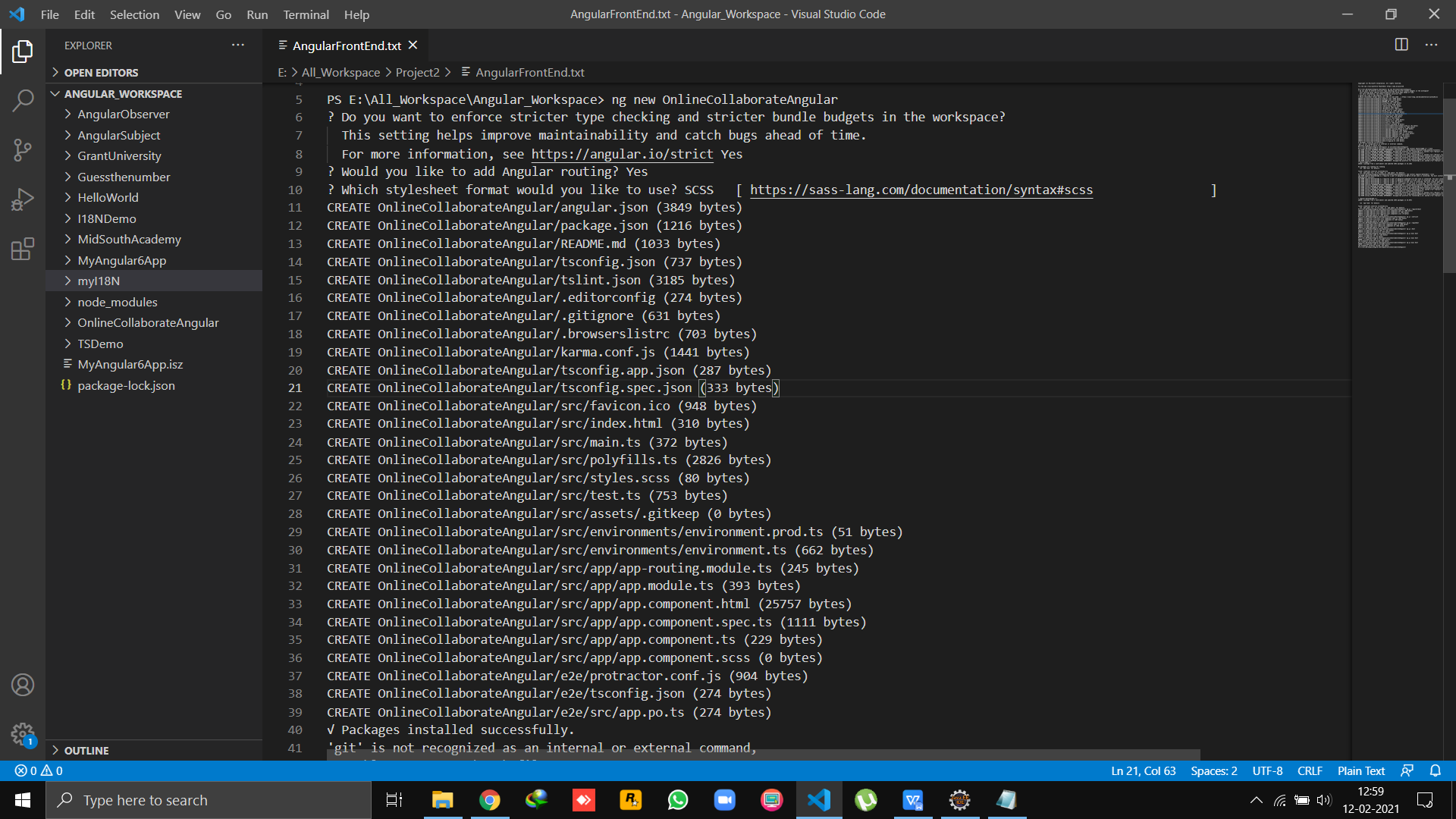
hibernateProperties.put("hibernate.show\_sql","true");

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

sessionFactory.setHibernateProperties(hibernateProperties);

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

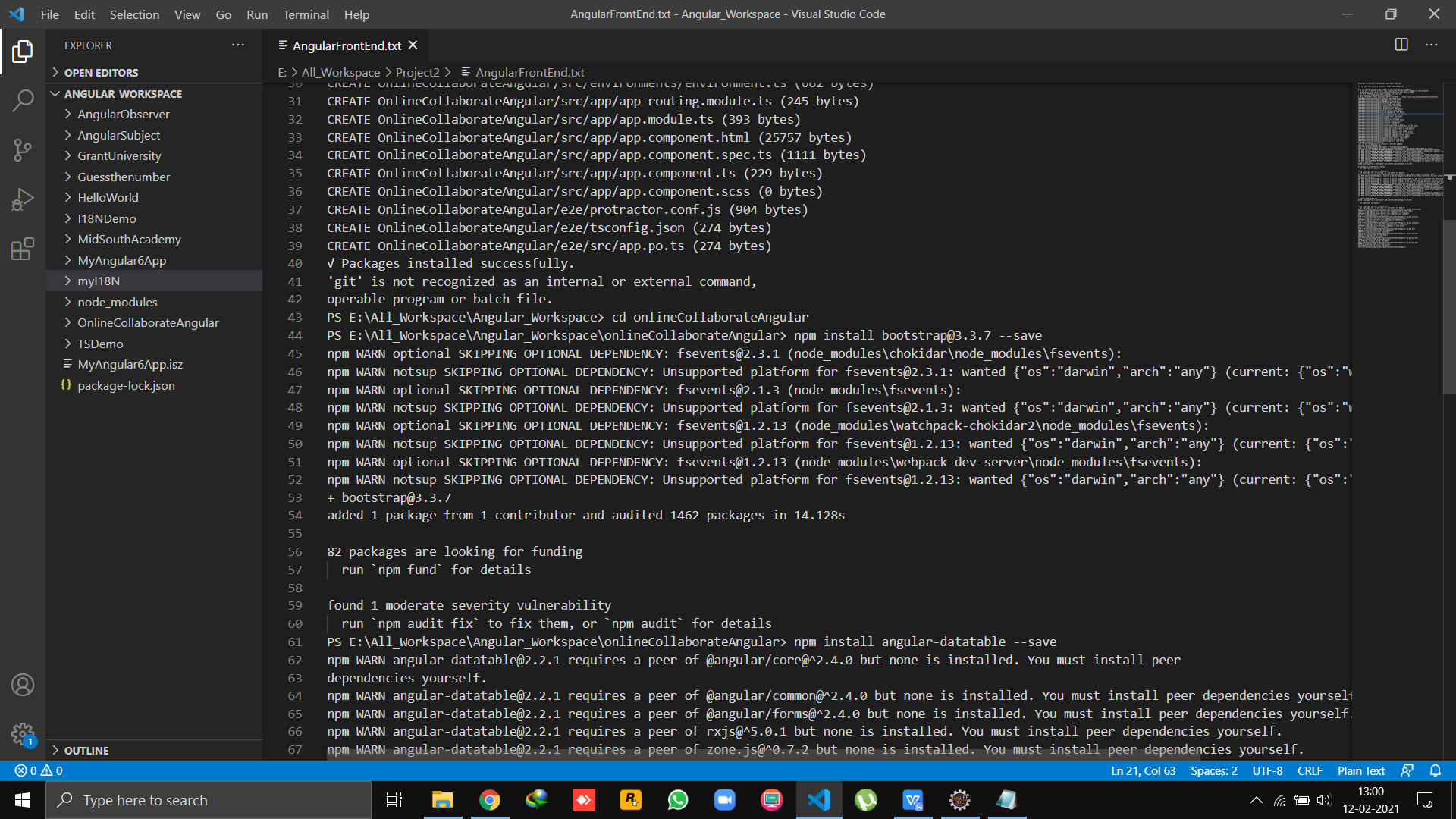
ng new OnlineCollaborateAngular



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](mailto:bootstrap@3.3.7) --save***



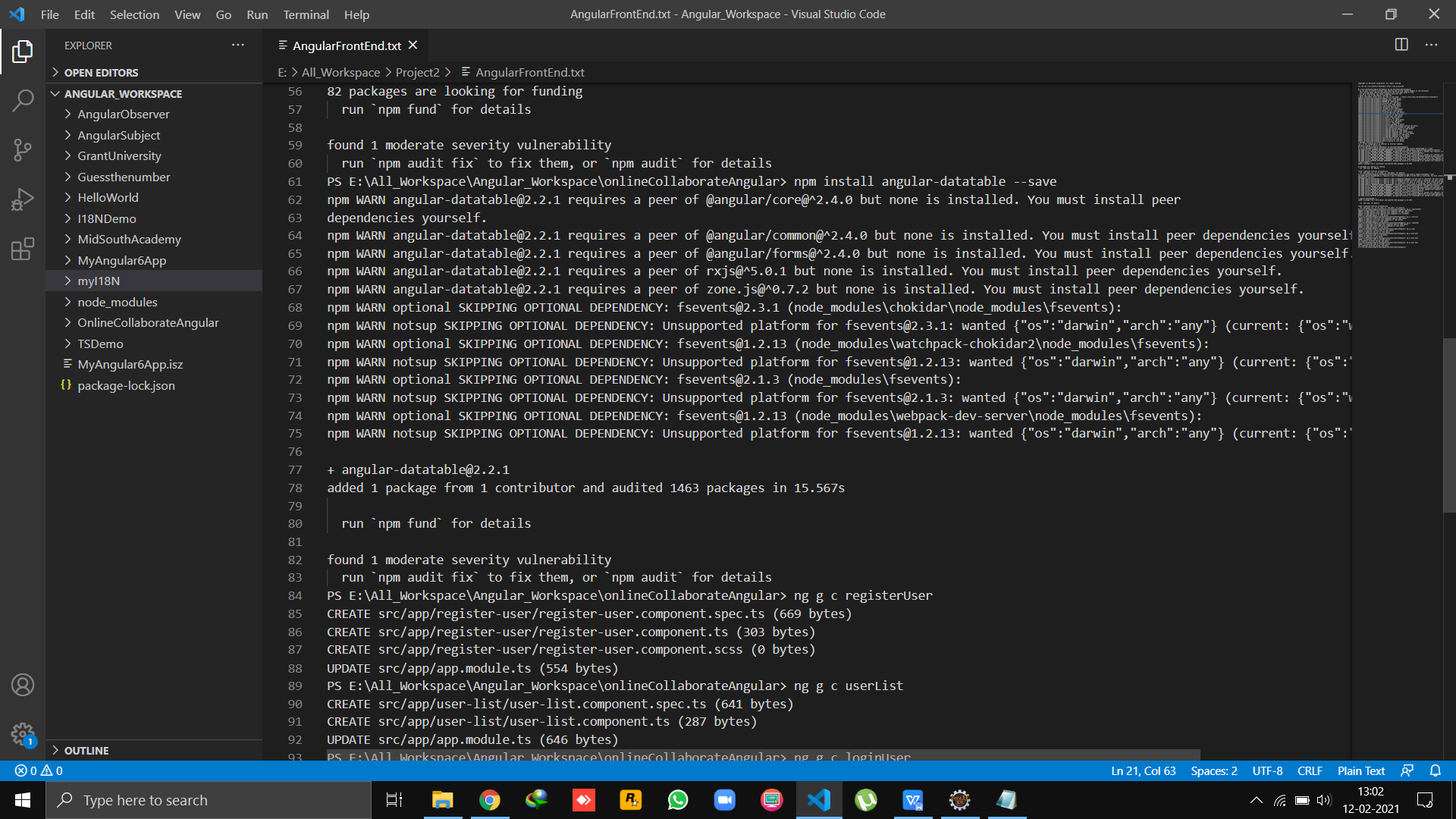
13.Now, include the following code in the style.css file.

@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***

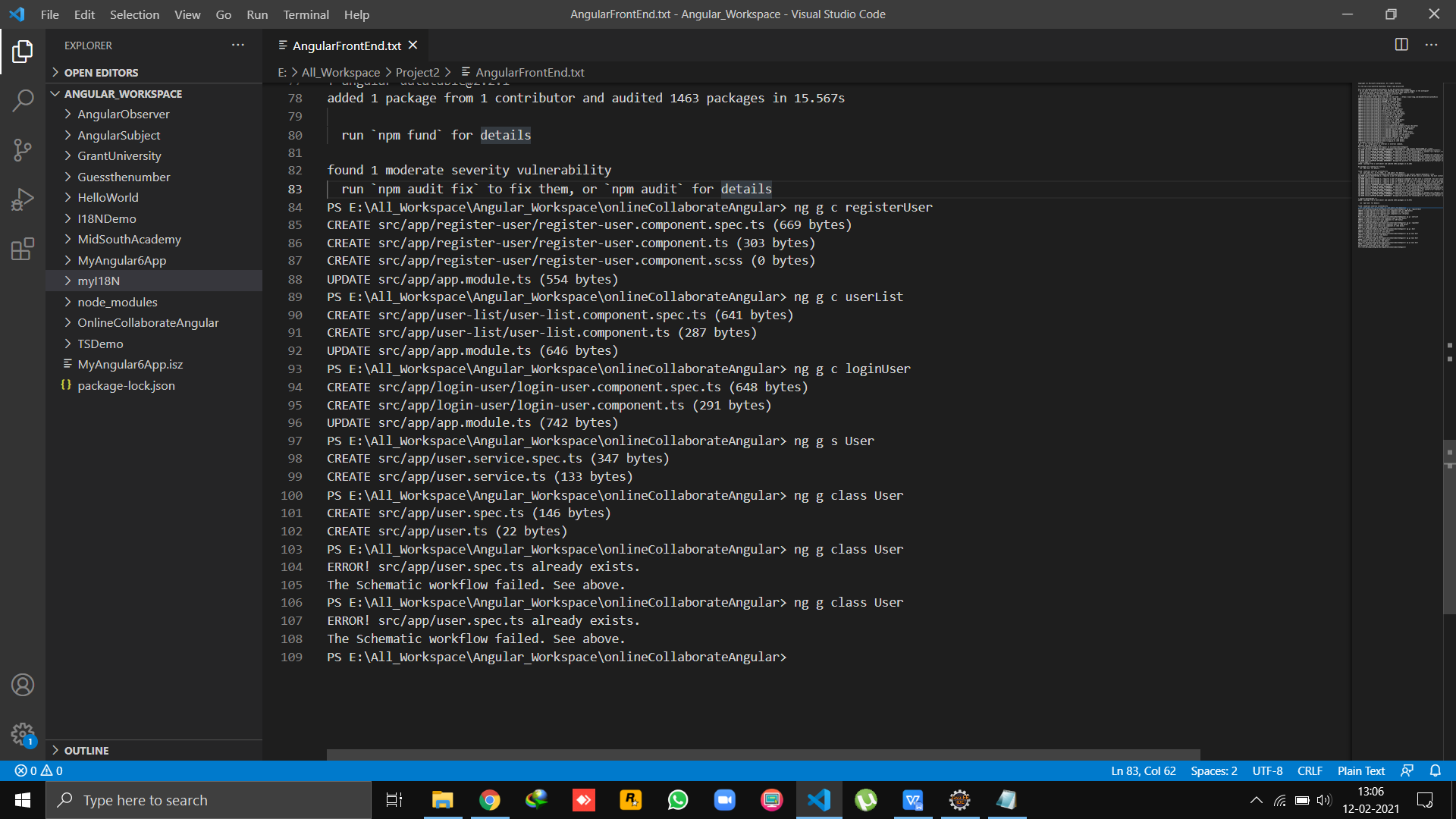


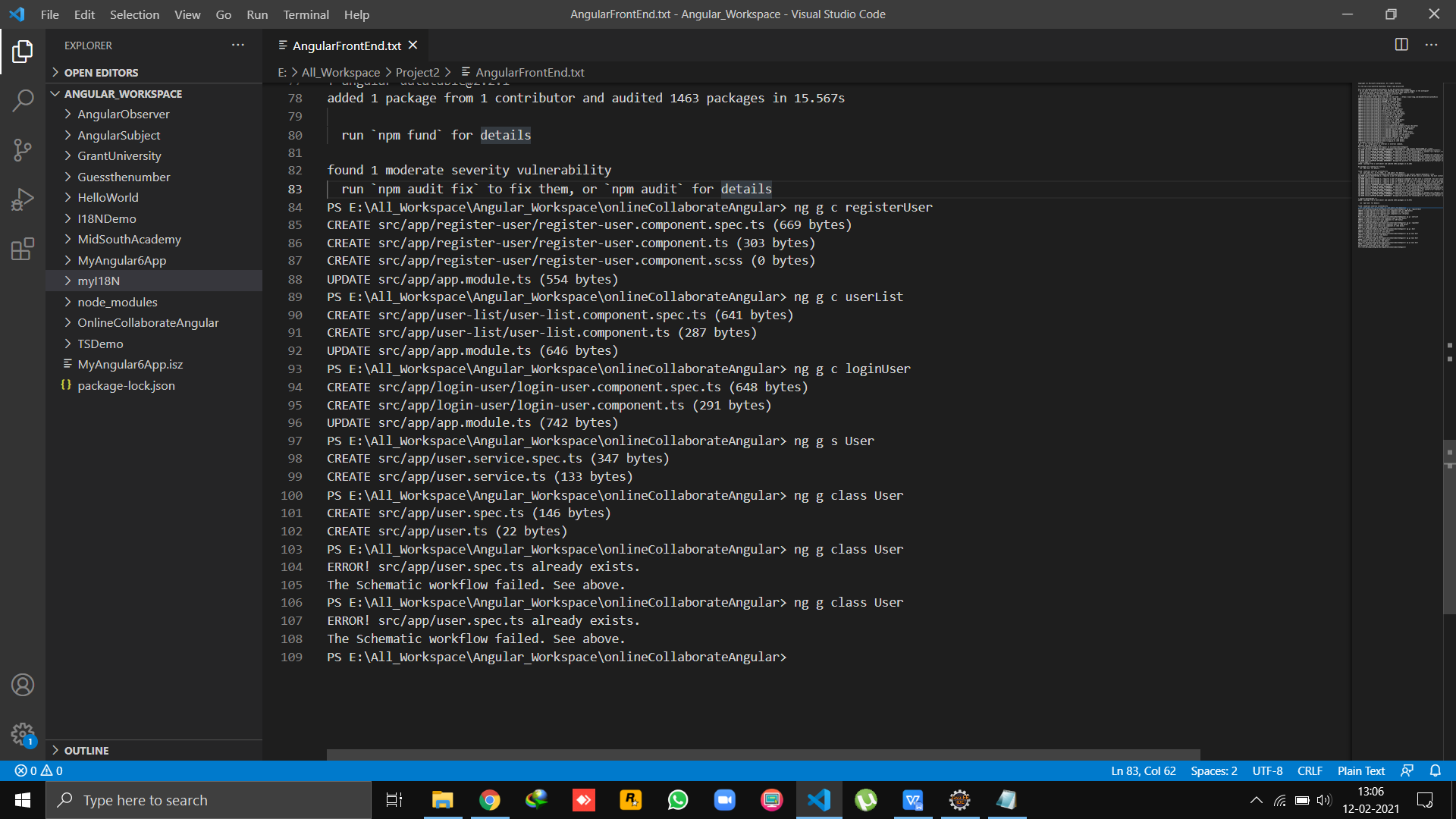
1. It is required to include **DataTablesModule** in imports array of **app.module.ts** file.
2. Generate Components  
   Open the project in visual studio and then use the following command to generate Angular components:

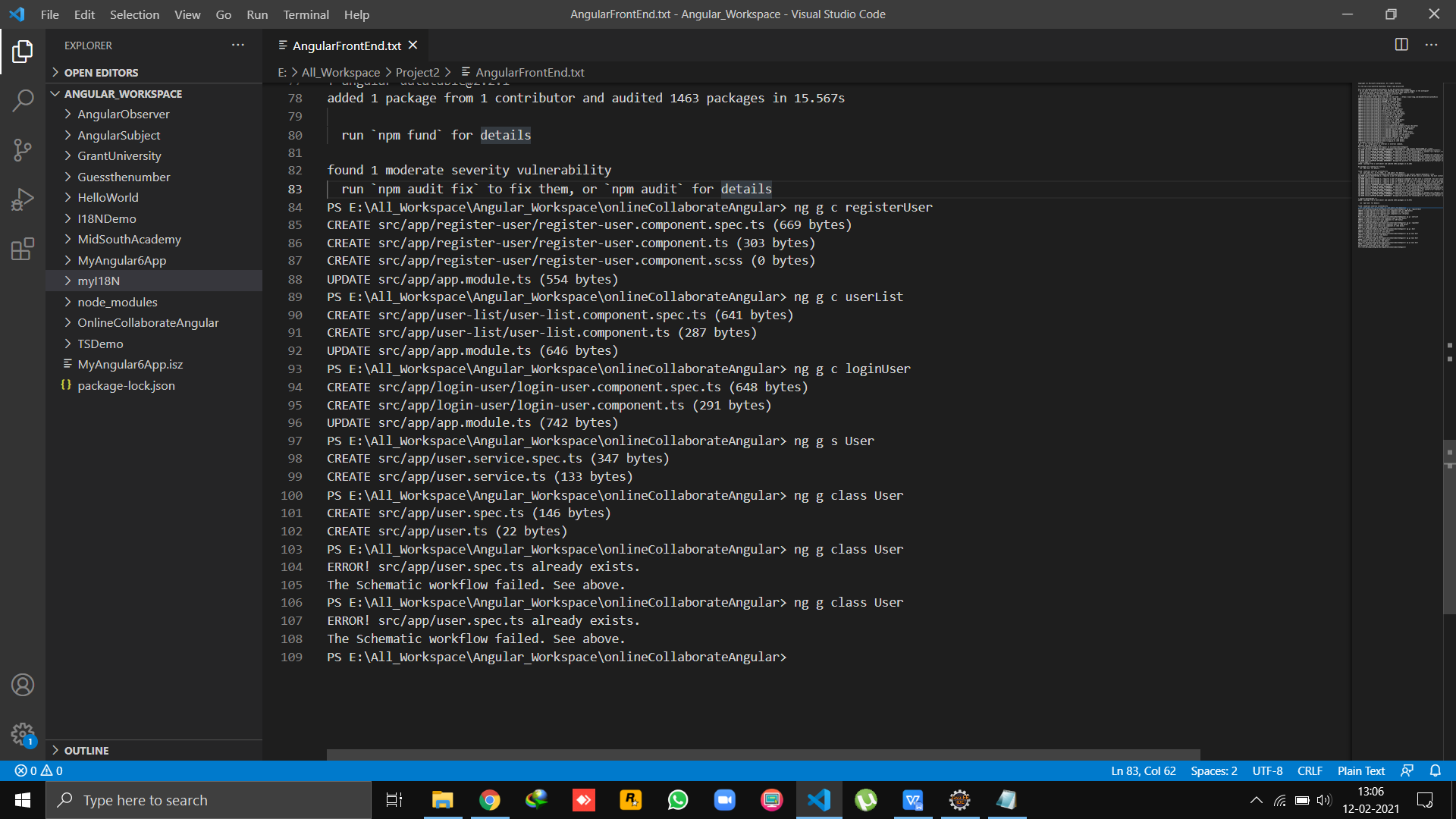
ng g c registerUser

ng g c userList

ng g c loginUser

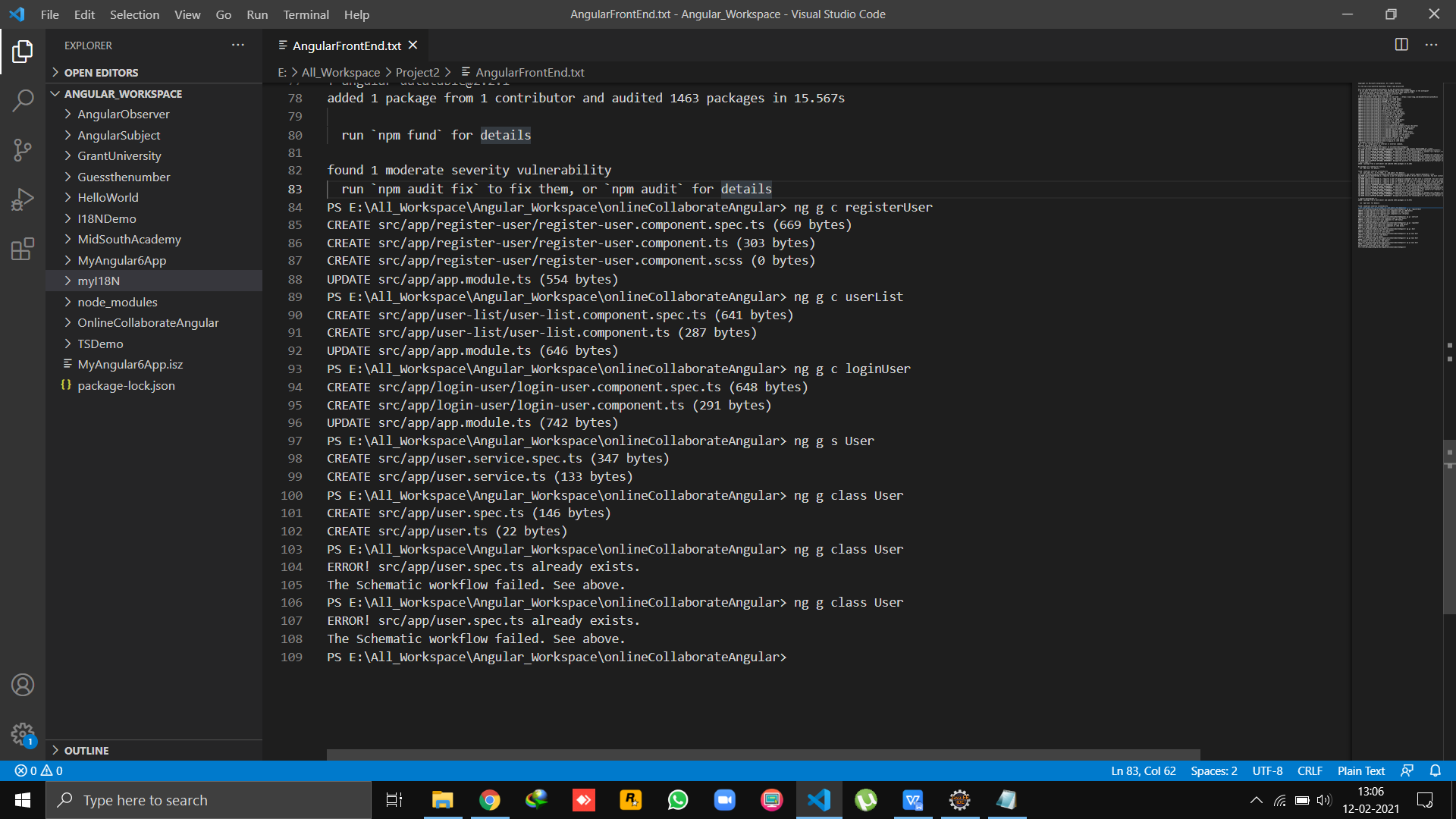






1. Let's also create a service class by using the following command: -

ng g s User



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

//app.module.ts

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

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

import { 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 { NavComponent } from './nav/nav.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';

@NgModule({

declarations: [

AppComponent,

RegisterUserComponent,

UserListComponent,

LoginUserComponent,

NavComponent,

UserProfileComponent,

BlogListComponent,

ActiveUserComponent,

HomeComponent

],

imports: [

BrowserModule,

AppRoutingModule,

FormsModule,

ReactiveFormsModule,

HttpClientModule,

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';

const routes: Routes = [

{ path: '', redirectTo: 'login-user', pathMatch: 'full' },

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

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

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

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

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

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

];

@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)

<app-home>

</app-home>

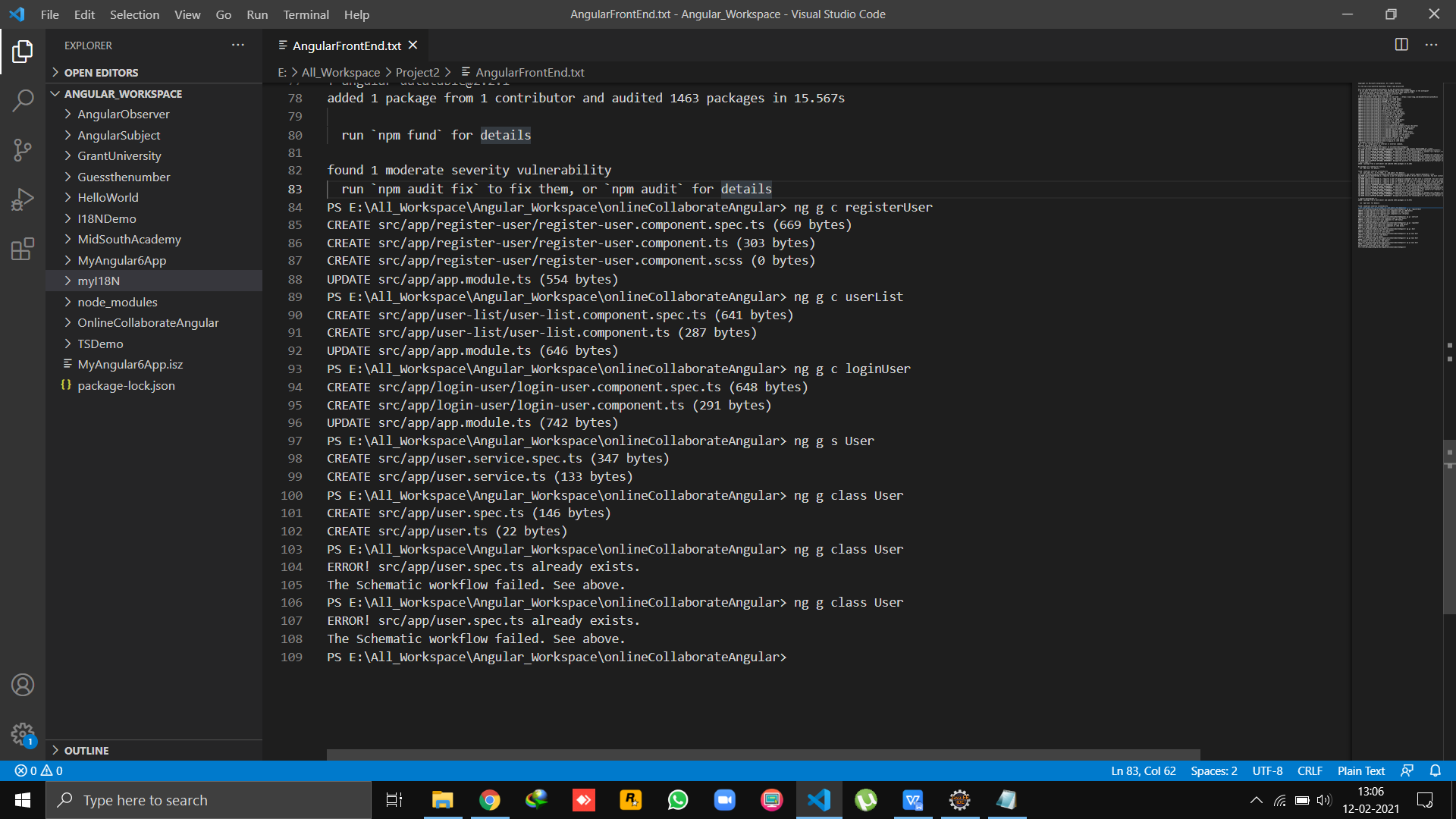
<div class="main-content">

</div>

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);

}

}

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();

}

}

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>

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!: Observable<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>

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

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';

@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) { }

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;

console.log(this.currentUser.firstName);

}

else {

console.log("Object Empty");

}

},

error => console.log(error)

)

}

get Username() {

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

}

get Password() {

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

}

}

Login-user.component.html

<div class="main">

<form [formGroup]="loginform" action="#" #validateuser="ngForm" (ngSubmit)="validateUser()" class="form1">

<p class="sign" align="center">Sign in</p>

<input type="text" class="form-control" name="username" required placeholder="Username" formControlName="username">

<input type="password" class="form-control" name="password" required placeholder="Password" formControlName="password">

<button type="submit" class="submit" align="center">Login</button>

</form>

</div>

Activate User

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!: Observable <User[]>;

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.lastName}}</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>

Navigatore

Nav.component.ts

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

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

import { Observable } from 'rxjs';

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

@Component({

selector: 'app-nav',

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

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

})

export class NavComponent {

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

.pipe(

map(result => result.matches),

shareReplay()

);

constructor(private breakpointObserver: BreakpointObserver) {}

}

Nav.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="login-user" class="nav-link" class="btn-primary active" role="button">Login</a>

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

<a mat-list-item routerLink="user-profile" class="nav-link" class="btn-primary active" role="button">User 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 style="font-style: italic;">OnlineCollaborate</span>

</mat-toolbar>

</mat-sidenav-content>

</mat-sidenav-container>

Home

Home.component.ts

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

@Component({

selector: 'app-home',

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

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

})

export class HomeComponent implements OnInit {

constructor() { }

ngOnInit(): void {

}

}

Home.component.html

<div class="main-container">

<div class="main">

<header>

<div class="overlay">

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

<br>

<br>

<br>

<a href="" class="btn">Sign / SignUp</a>

</div>

</header>

</div>

</div>