**Online Collaborate**

Name-Joyasree Mondal

Reg No-R2011902000104

NIIT

**Online Collaboration**

Batch Code: S210167

Start Date:11/11/20

End Date:15/03/20

Name of the Coordinator: Lopamudra Bera

Names of the Developers: Joyasree Mondal

Date of Submission:15/03/2020

**NIIT**

**CERTIFICATE**

This is to certify that this report, titled **Online Collaborate** embodies the original work done by **Joyasree Mondal**, in partial fulfillment of his course requirement at NIIT.

Coordinator: Joyasree Mondal

**ACKNOWLEDGEMENT**

I express my profound sense of reverence to my coordinator Ms. Lopamudra Bera, for her constant guidance, support ,motivation and untiring help during the course of my project work .she has given us enough freedom during our project, and has always been nice to us. i must acknowledgement and thank the blessing ,protecting and guiding me thought this period.

**ABSTRACT**

This project will mainly use for professional ,allowing registered users to post blogs and their information is organized and easily accessible to all registered users.

Users will register themselves in the register from and fill their professional details. And Admin can be able to mange all the process from activating the register user to approving a blog posted the user.

**CONFIGURATION**

* Hardware :
* 4GB RAM,1TB HARD DISK, INTEL I3 10th GENERATION PROCESSOR
* Operating system:
* WINDOWS 10,x64
* Software:
* Eclipse IDE
* Java
* Spring Boot
* Spring Hibernate
* MySQL
* Angular
* Html
* Typescript
* CSS
* jQuery

**TABLE OF CONTENTS**

**Chapter 1 -introduction**

* 1. Aim
  2. Objectives

**Chapter 2-project Requirement Specification**

2.1Statement of requirement

**Chapter 3 -project Analysis**

3.1 project plan

3.2 Business process Model

3.3 Software Requirement Specification

3.4 Dispatcher Servlet Diagram

3.5 High Level Use Case Diagram

**Chapter 4 -Project Design**

4.1 Angular Lifecycle

4.2 User interface Design

4.3 Database Structure

4.4 Data Model

4.5 project Structure

4.6 Low Level Use Case Diagram

4.7 Systems Input and Output Design

**Chapter 5 -Angular-framework**

5.1 Front-end

5.2 personal Observation

Challenges

Observation

Reference

Appendix

**AIM**

The main objective of the Online Blogging System is to manage the details of Blogs, Topic, Idea, Content ,Entries. It manages all the information about Blogs, Views, Entries, Blogs. The project is totally built at administrative end and thus only the administrator is guaranteed

**OBJECTIVE**

The project will mainly use for professional networking, allowing registered users to blogs and their information and its easily collaborate with all other register.

User will register themselves in the application and fill application and fill in their personal details. And uses can be able to see their profile.

**STATEMENT REQUIRMENTS**

|  |
| --- |
| PROJECT OBJECTIVES |
| |  |  | | --- | --- | | **TiTle** | **Online Collaborate** | | **Subtitle** | **Blog Management System** | | **Author(s)** | **Joyasree Mondal** | | **Author’s E-mail** | **Joyasree.niit.mondal@gmail.com** | | **Author’s phone** | **8513032171** | | **Description** | **Blogging** | | **version** | **1.o1** | |

Key Features of the application:

1.its has home page window that user/Admin can login/Register.

2.it has registration window to register for user register by themselves.

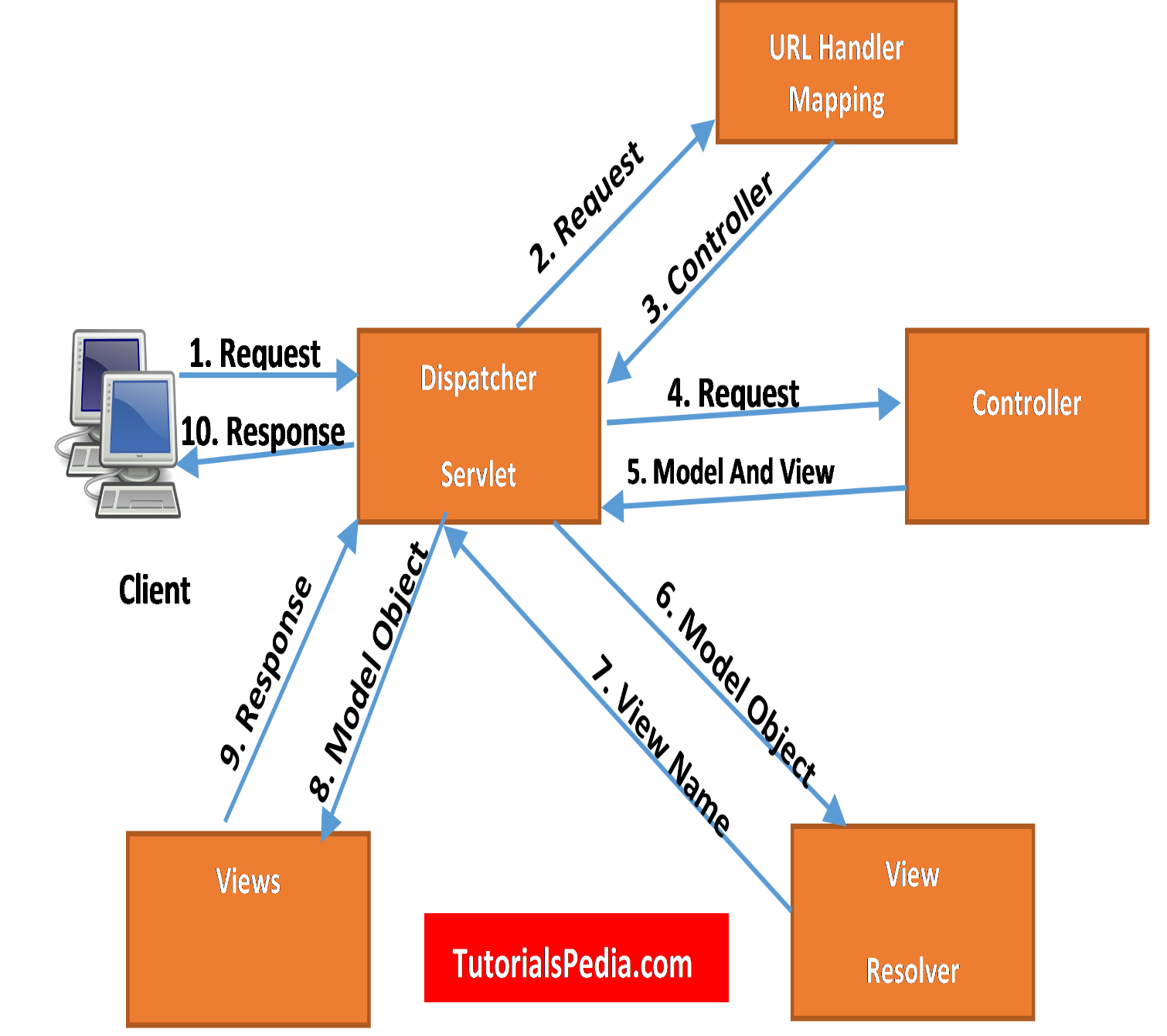
3.Application has the same login window for everyone and it detects by itself which designation you are in and give access accordingly.

4 Admin can able to activate and deactivate an users profile and blog their posted.

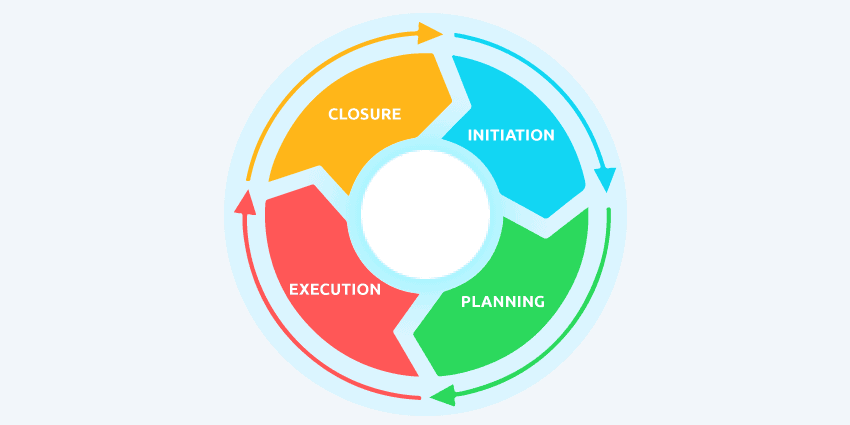
5.Admin can able to view all the user status and can update,delete.active or deactivate them.

6.user can add a posting on the their own blog and update or delete a blog while it is needed.

**DISPATCHER SERVLET DIAGRAM**

****

**PROJECT LIFE CYCLE MODEL**

****

The Initiation Phase: The initiation phase aims to define and authorize the project. Vision: Registration ,Authorization, Blogging

The Planning Phase: The purpose of this phase is to lay down a detailed strategy of how the project has to be performed and how to make it a success. Strategic Planning- overall approach to the project Implementation Planning-ways to apply the decisions

The Execution Phase: In this phase, the decisions and activities defined during the planning phase are implemented.

The Termination Phase: This is the last phase of any project, and it marks the official closure of

**SYSTEM ARCHITECTURE**

Online Collaboration

Home

Angular Component

(Login,Registration,user-list,blog-List,Service)

User

Admin

Presentation Layer

DATA ACESS LAYER

APPLICATION LaYER

Hibernate

Configuration

CRUD

Controller

Service

DATABASE

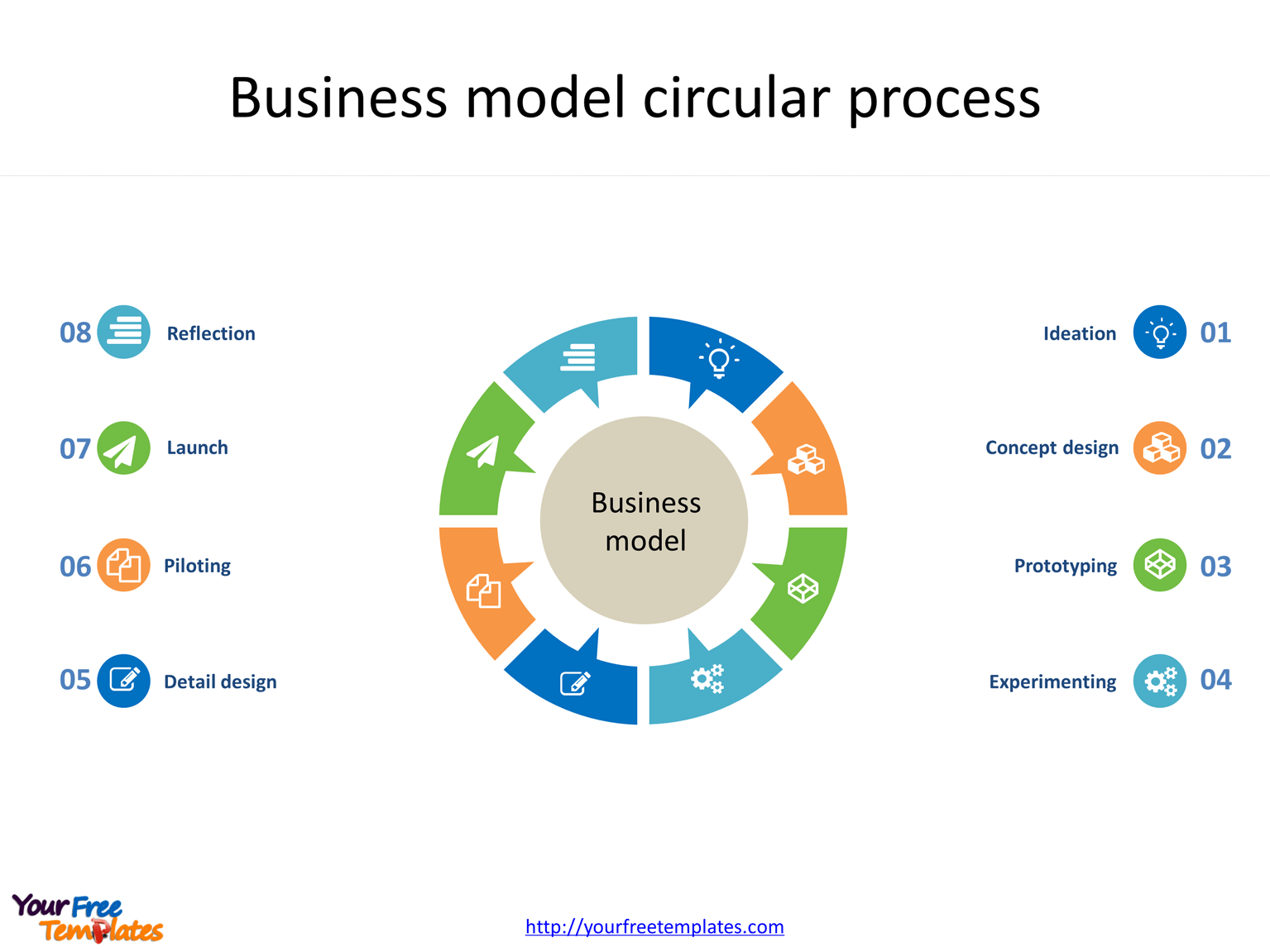
(Tables-User,Blog.BlogComments)

3-Tier Architecture 1. A Presentation Layer that sends content to browsers in the form of HTML/JS/CSS. This might leverage frameworks like React, Angular, Ember, Aurora, etc.

2. An Application Layer that uses an application server and processes the business logic for the application. This might be written in C#, Java, C++, Python, Ruby, etc. 3.

A Data Layer which is a database management system that provides access to application data. This could be MSSQL, MySQL,

**BUSINESS PROCESS MODEL**

****

Business process modeling (or) process modeling, is the analytical representation or put simply an illustration of an organization’s business processes. Modeling processes is a critical component for effective business process management.

Benefits of business process modeling:

* Gives everyone a clear understanding of how the process works
* Provides consistency and controls the process
* Identifies and eliminates redundancies and inefficiencies
* Sets a clear starting and ending to the

**SOWFTWARE REQUIREMNETS**

**1.Introduction**

1.1 Purpose

1.2 Scope

1.3 Definitions, acronyms and abbreviations

1.4 References

1.5 Overview

**2.General Description**

2.1 Product perspective

2.2 Product functions

2.3 User characteristics

2.4 Constraints

2.5 Assumption and dependencies

**3. Specific Requirements**

3.1 Functional requirements

3.2 Non-functional requirements

3.3 External interface requirements

3.4 Performance requirements

3.5 Design constraints

3.6 Attributes

3.7 Other requirements

**4. Appendices**

**5. Index**

**HIGH LEVEL USER CASE DIAGRAMS**

Vaildation

Sign in

Home

Nav

User

Admin

Sign up

Registration From

User

Admin

Curd

Darabase

Hibernate config

FrontEnd

**LOW LEVEL USER CASE DIAGRAMS**

Database Layer

Application Layer

User interface Layer

**DATABASE MODEL**

BlogComments

UserId int not null auto\_ increment

firstName varchar

lastName varchar

username varchar

PassWord varchar

Email varchar

Role varchar

Status varchar

inOnline varchar

enabled varchar

User

Blog

blogId int not null auto\_incerment

blogTitle varchar

blogContent varchar

blogPosted varchar

status varchar

noOfLikes varchar

noComments int

noOfViews int

userId int

username varchar

blogComments int not auto\_increment

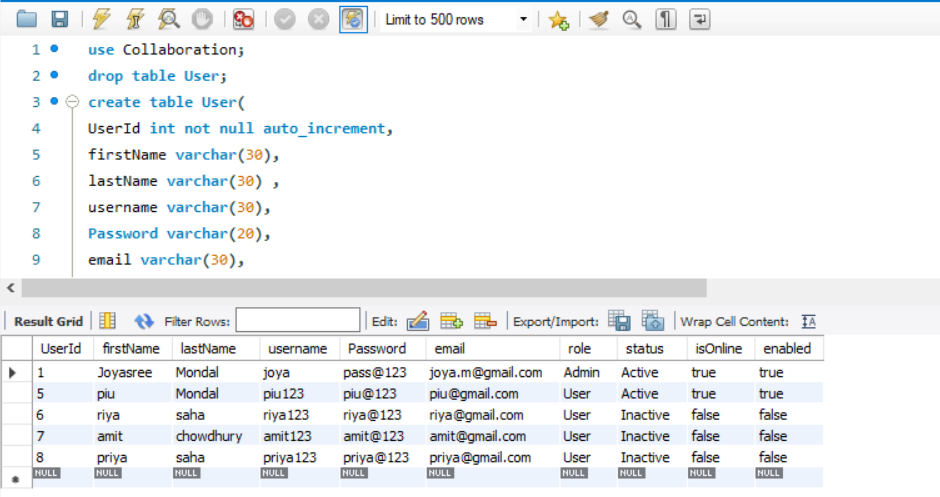
userId int

username varchar

userProfileId varchat

**DATA MODEL**

**User Table:**

****

Use Collaboration;

**Create Tables**

1.User Tables:

create table User(

UserId int not null auto\_increment,

firstName varchar(30),

lastName varchar(30) ,

username varchar(30),

Password varchar(20),

email varchar(30),

role varchar(10),

status varchar(10),

isOnline varchar(5),

enabled varchar(5),

primary key(UserId)

);

Select \* from user

**Create table Blog**

create table Blog(

blogId int not null auto\_increment,

blogTiTle varchar(30),

blogContent varchar(200),

blogPosted datetime,

status varchar(10),

noOfLikes int,

noComments int,

noOfViews int,

userId int,

username varchar(20),

primary key(BlogId)

);

Select \* from blog

**Create table BlogComments**

create table BlogComments(

blogCommentId int not null auto\_increment,

userId int ,

username varchar(20),

userProfileId varchar(20),

title varchar(30),

noOfLikes int,

blogcomment varchar(20),

currentDate date,

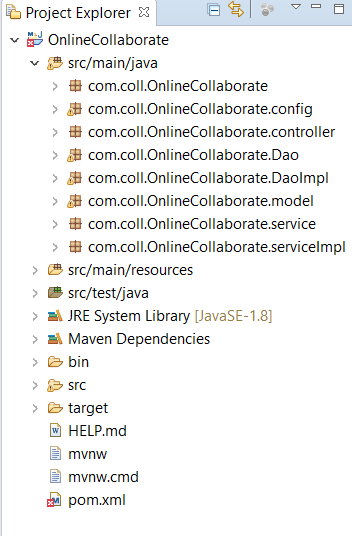
blogId int,

primary key(blogCommentId)

);

Select \* from blogComments

**PROJECT MODEL**

****

This is the project model that I have created on Eclipse IDE.The Structure of the project is presents. There are will get to see different kind of package like config, controller, service, serviceImpl, dao, daoImpl, model ets.these are classes which can handle the entire backend proess of the project structure**.**

**Code:**

* Create a Spring Boot project named online Collaborate with web, Spring data JPA, Spring Dev Tools and MySQL server driver package. Extract that project.
* Import the project in Eclipse.

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>

.

* Open package com. coll. Online Collaborate. config.
* Configuration file as 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.OnlineCollaborate"),

@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;

}

}

* Create model class:

1.User

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;

}

}

* Create Interface:

2.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 getUserByUserName(String Username);

User validateUser(User user);

boolean addUser(User user);

boolean updateUser(User user);

boolean deleteUser(int userId);

boolean deactiveUser(int userId);

List<User> getAllDeactiveUser();

boolean activeUser(int userId);

boolean updateUserProfile(String file,Integer userId);

}

* create DaoImpl class

3.UserDaoImpl

package com.coll.OnlineCollaborate.DaoImpl

import java.util.List;

import javax.persistence.Query;

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.IUserDao;

//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 getUserByUserName(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 ex){

ex.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");

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 List<User> getAllDeactiveUser() {

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

}

@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;

}

}

}

* Create service interface

4.IUserService Interface

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 getUserByUserName(String Username);

User validateUser(User user);

boolean addUser(User user);

boolean updateUser(User user);

boolean deleteUser(int userId);

boolean deactiveUser(int userId);

List<User> getAllDeactiveUser();

boolean activeUser(int userId);

boolean updateUserProfile(String file,Integer userId);

}

* Create serviceImpl

5.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.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) {

return userDao.userListbyStatus(status);

}

@Override

public List<User> getAllUsers() {

return userDao.getAllUsers();

}

@Override

public User getUserById(int UserId) {

return userDao.getUserById(UserId);

}

@Override

public User getUserByUserName(String Username) {

return userDao.getUserByUserName(Username);

}

@Override

public User validateUser(User user) {

return userDao.validateUser(user);

}

@Override

public boolean addUser(User user) {

return userDao.addUser(user);

}

@Override

public boolean updateUser(User user) {

return userDao.updateUser(user);

}

@Override

public boolean deleteUser(int userId) {

return userDao.deleteUser(userId);

}

@Override

public boolean deactiveUser(int userId) {

return userDao.deactiveUser(userId);

}

@Override

public boolean updateUserProfile(String file, Integer userId) {

return userDao.updateUserProfile(file, userId);

}

@Override

public List<User> getAllDeactiveUser() {

return userDao.getAllDeactiveUser();

}

@Override

public boolean activeUser(int userId) {

return userDao.activeUser(userId);

}

}

* Create controller

6.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 UserCotroller {

@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> AllDeactiveUsers(){

return userService.getAllDeactiveUser();

}

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

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

return userService.activeUser(userId);

}

@PostMapping("validate-user")

public User validateUser(@RequestBody User user) {

return userService.validateUser(user);

}

}

* Create model class

1.Blog:

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 blogTitle, blogContent;

LocalDate blogPosted;

String status;

int noOfLikes, noOfComments,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 getBlogTitle() {

return blogTitle;

}

public void setBlogTitle(String blogTitle) {

this.blogTitle = blogTitle;

}

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 getNoOfComments() {

return noOfComments;

}

public void setNoOfComments(int noOfComments) {

this.noOfComments = noOfComments;

}

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;

}

}

* Create interface

2.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(Blog blog);

}

* Create BolgDaoImpl

3.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;

@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 blog) {

try

{

sessionFactory.getCurrentSession().delete(blog);

return true;

}

catch(Exception ex)

{

ex.printStackTrace();

return false;

}

}

}

* create BlogService:

4.IBlogService:

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);

Blog getBlogById(int blogId);

boolean addBlog(Blog blog);

boolean updateBlog(Blog blog);

boolean deleteBlog(Blog blog);

}

* Create BlogServiceImpl

5.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.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(Blog blog) {

return blogDao.deleteBlog(blog);

}

}

* Create BlogController

6.BlogController class

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}")

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

{

return blogService.deleteBlog(blog);

}

@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);

}

}

* Create BlogComments model class

1.model class:

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 extends DomainResponse 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;

}

}

* Create IBlogcommentsDao Interface

2.IBlogComments:

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> getAllBlogsComments();

List<BlogComments> getCommentsbyBlog(int blog);

List<BlogComments> getBlogsCommentsByStatus(String status);

BlogComments getBlogCommentsById(int blogcommentsId);

boolean addBlogComments(BlogComments blogComments);

boolean updateBlogComments(BlogComments blogComments);

boolean deleteBlogComments(BlogComments blogComments);

}

* Create BlogComments DaoImpl

3.BlogCommentsDaoImpl

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.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> getAllBlogsComments() {

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

}

@Override

public List<BlogComments> getBlogsCommentsByStatus(String status) {

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

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

return query.getResultList();

}

public List<BlogComments> getCommentsbyBlog(int blog){

return null;

}

@Override

public BlogComments getBlogCommentsById(int blogcommentsId) {

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

}

@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 blogcomments) {

try

{

sessionFactory.getCurrentSession().delete(blogcomments);

return true;

}

catch(Exception ex)

{

ex.printStackTrace();

return false;

}

}

}

* Create IBlogCommentsService Interface

4.IBlogCommentsService:

package com.coll.OnlineCollaborate.service;

import java.util.List;

import com.coll.OnlineCollaborate.model.BlogComments;

public interface IBlogCommentsService {

List<BlogComments> getAllBlogsComments();

List<BlogComments> getBlogsCommentsByStatus(String status);

List<BlogComments> getUsersBlogsComments(int blogcommentsId);

List<BlogComments> mainList();

BlogComments getBlogCommentsById(int blogcommentsId);

boolean addBlogComments(BlogComments blogcomments);

boolean updateBlogComments(BlogComments blogcomments);

boolean deleteBlogComments(BlogComments blogcomments);

}

* Create BlogCommentsServiceDaoImpl class

5.BlogCommentsServiceDaoImpl

package com.coll.OnlineCollaborate.serviceImpl;

import java.util.List;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.coll.OnlineCollaborate.model.BlogComments;

import com.coll.OnlineCollaborate.service.IBlogCommentsService;

@Service

@Transactional

public class BlogCommentsServiceImpl implements IBlogCommentsService{

@Override

public List<BlogComments> getAllBlogsComments() {

// TODO Auto-generated method stub

return null;

}

@Override

public List<BlogComments> getBlogsCommentsByStatus(String status) {

// TODO Auto-generated method stub

return null;

}

@Override

public List<BlogComments> getUsersBlogsComments(int blogcommentsId) {

// TODO Auto-generated method stub

return null;

}

@Override

public List<BlogComments> mainList() {

// TODO Auto-generated method stub

return null;

}

@Override

public BlogComments getBlogCommentsById(int blogcommentsId) {

// TODO Auto-generated method stub

return null;

}

@Override

public boolean addBlogComments(BlogComments blogcomments) {

// TODO Auto-generated method stub

return false;

}

@Override

public boolean updateBlogComments(BlogComments blogcomments) {

// TODO Auto-generated method stub

return false;

}

@Override

public boolean deleteBlogComments(BlogComments blogcomments) {

// TODO Auto-generated method stub

return false;

}

}

* Create BlogCommentsController class

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.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.getAllBlogsComments();

}

@DeleteMapping("delete-blogcomments/{blogcomments}")

public boolean deleteBlogComments(@PathVariable("blogcomments") BlogComments blogcomments)

{

return blogcommentsService.deleteBlogComments(blogcomments);

}

@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);

}

}

**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) {

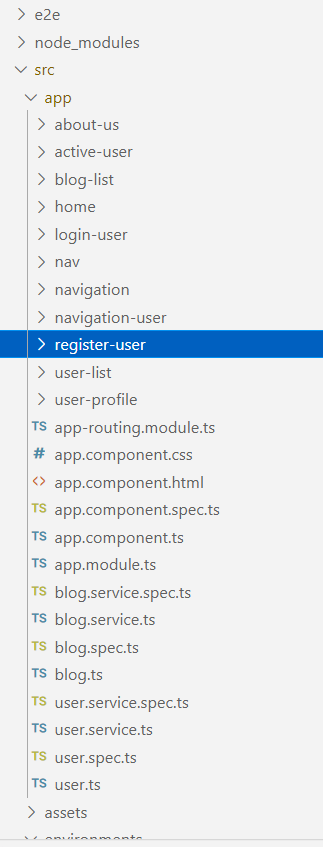
SpringApplication.run(OnlineCollaborateApplication.class, args);

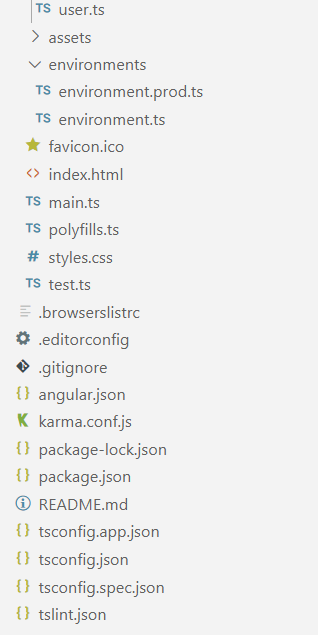
}

}

* + Right click OnlineCollaborateApplication
  + Click Run as
  + Click JavaApplication

**Front End Procedure**

****



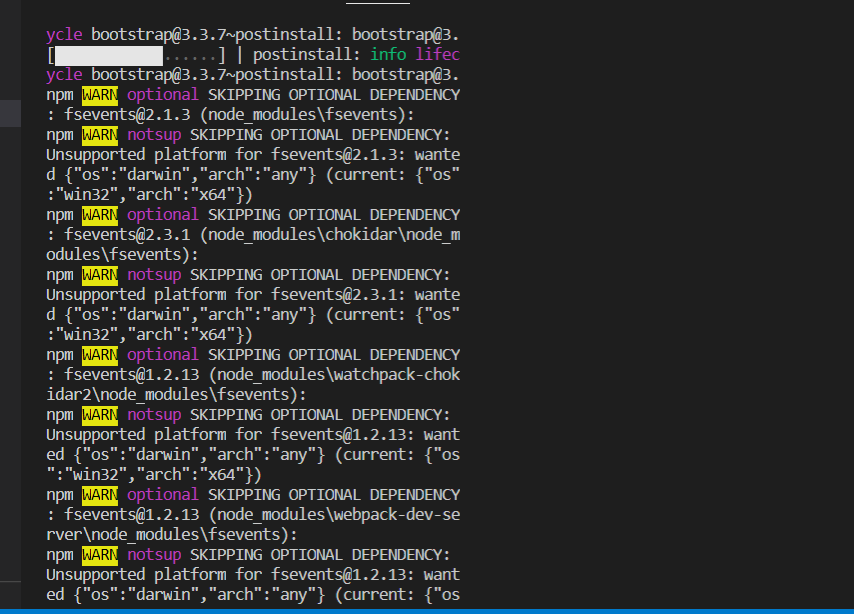
* Open visual Studio code create new project inside Angular\_Workspaces . set that project name as Online Collaborate

Ng new OnlineCollaborateAngular

* Install bootstrap:

Use the following command to install bootstrap:

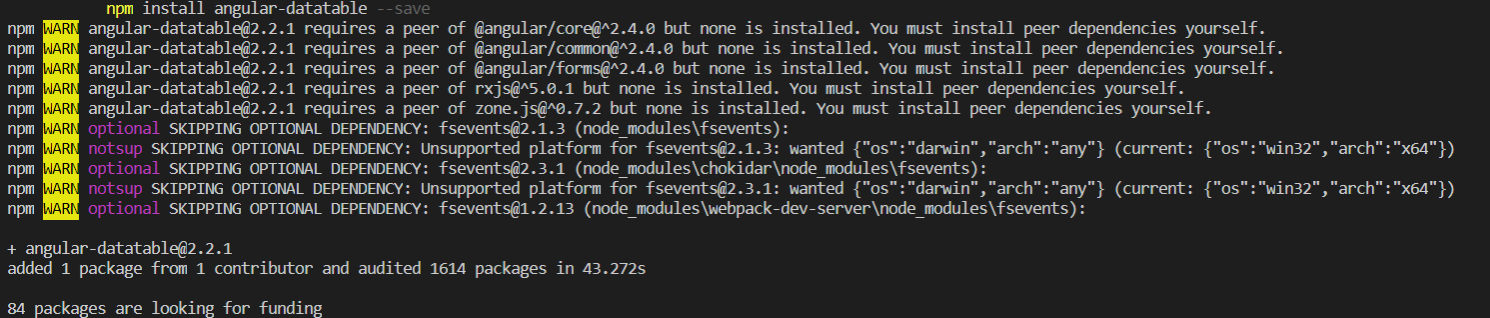
Npm install [bootstrap@3.3.7](mailto:bootstrap@3.3.7)



* Install Angular Data Table

Use the following commands:

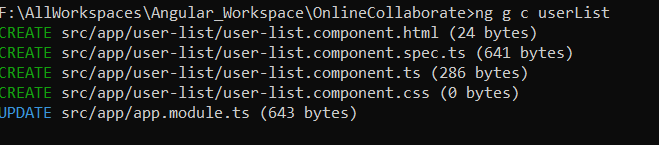
Npm install angular- datatable –save



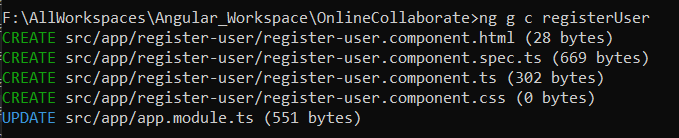
* Generate Components:

Open the project in visual studio than use the following commands to generate Angular components:

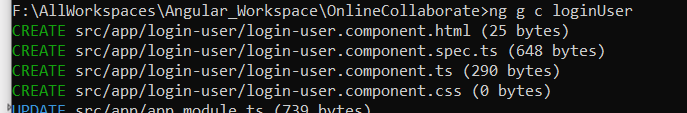
**Ng g c user-list**



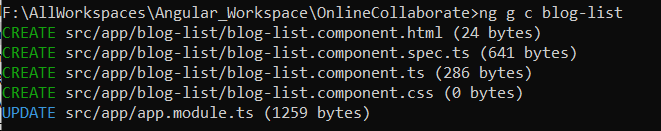
**Ng g c register-user**

****

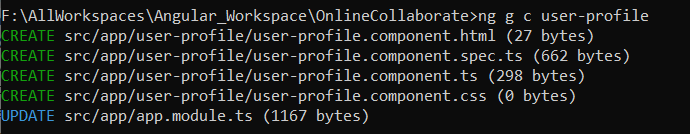
**Ng g c login -user**



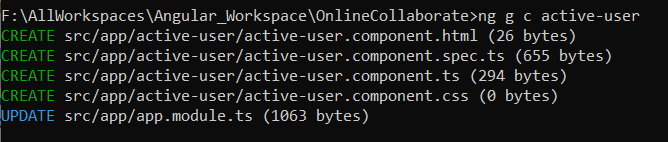
**Ng g c blog-list**



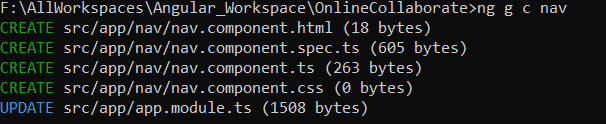
**Ng g c user-profile**

****

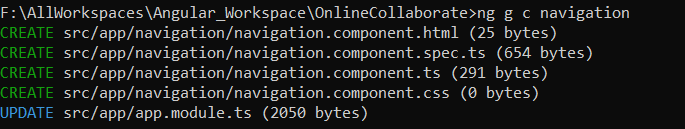
**Ng g c active-user**

****

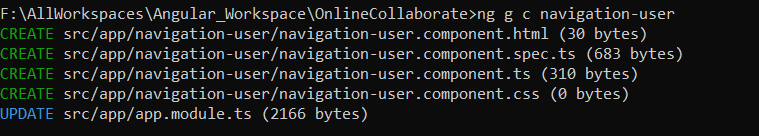
**Ng g c nav**

****

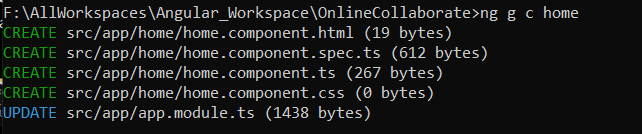
**Ng g c navigation**

****

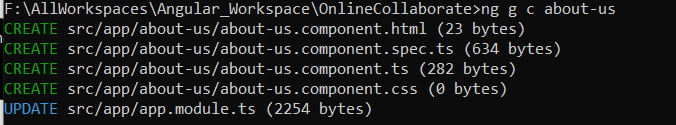
**Ng g c navigation-user**

****

**Ng g c home**

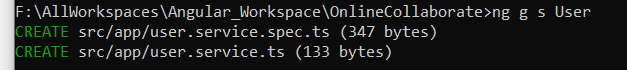
****

**Ng g c about-us**

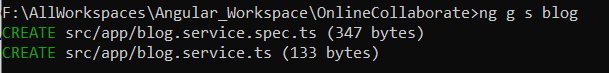
****

* **Create a service class using the following commands**

**Ng g s user**

****

**Ng g s blog**

****

* Edit the app. module. ts file:
* Import Routing
* Import ReactiveFormsModule
* Import HttpModule
* Import service class

**App. module. ts file:**

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

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

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

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

import {DataTablesModule} from 'angular-datatables';

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

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

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

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

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

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

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

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

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

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

@NgModule({

  declarations: [

    AppComponent,

    RegisterUserComponent,

    UserListComponent,

    LoginUserComponent,

    ActiveUserComponent,

    UserProfileComponent,

    BlogListComponent,

    HomeComponent,

    NavComponent,

    NavigationComponent,

    NavigationUserComponent,

    AboutUsComponent,

  ],

  imports: [

    BrowserModule,

    AppRoutingModule,

    FormsModule,

     ReactiveFormsModule,

    HttpClientModule,

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

  ],

  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }

**App routing.module.ts file**

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

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

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

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

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

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

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

const routes: Routes = [

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

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

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

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

  ];

@NgModule({

  imports: [RouterModule.forRoot(routes)],

  exports: [RouterModule]

})

export class AppRoutingModule { }

**app.components.html**

<router-outlet>

  </router-outlet>

**App.component.css:**

main-content{

    left: 100px;

    max-width: 70;

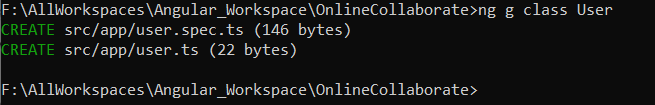
    background-color:aqua;

}

**Create the user.ts class**

* Lets create user.ts following commands:

Ng g class user



**User.ts file:**

export class User{

    userId:number;

    firstName:String;

    lastName:String;

    username:String;

    password:String;

    confirm\_password:String;

    email:String;

    role:String;

    status:String;

    isOnline:String;

    enabled:String;

}

**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');

  }

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

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

  }

  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> {

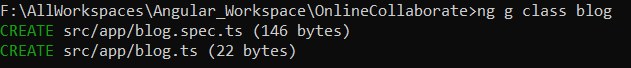
        return this.http.post(`${this.baseUrl}/logout-user/${userId}`, {responseType: 'text'});

      }

}

* Create the blog.ts class:

Ng g blog



**Blog.ts file:**

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 file:**

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);

 }

}

**User-list.components.ts:**

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

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

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

import { User } from '../user';

import {Observable,Subject} from "rxjs";

import {DataTablesModule} from 'angular-datatables';

@Component({

  selector: 'app-user-list',

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

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

})

export class UserListComponent implements OnInit {

  constructor(private userservice:UserService) { }

  usersArray: any=[];

  dtOptions: DataTables.Settings={};

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

  users : Observable<User[]>;

  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(id: number){

    this.userservice.deleteUser(id)

    .subscribe(

      data=> {

        console.log(data);

        this.deleteMessage=true;

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

          this.users=data

        })

      },

      error=> console.log(error));

  }

  updateUser(id: number){

    this.userservice.getUser(id)

    .subscribe(

      data=>{

        this.userlist=data;

        console.log(this.userlist);

      }),

      error=> console.log(error);

  }

  userupdatefrom=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(),

  });

  update(user){

    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(error));

  }

  get firstName(){

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

  }

  get lastName(){

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

  }

  get userName(){

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

  }

  get password(){

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

  }

  get email(){

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

  }

  get role(){

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

  }

  get status(){

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

  }

  get isOnline(){

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

  }

  get enabled(){

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

  }

  get userId(){

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

  }

  changeisUpdate(){

    this.isupdated=false;

  }

}

**User-list components.html**:

<div class="panel panel-default">

    <div class="panel-heading">

        <h1 style="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>password</th>

                    <th>email</th>

                    <th>role</th>

                    <th>status</th>

                    <th>isOnline</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.password}}</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>

                      <button (click)="updateUser(user.userId)" class='btn btn-info'

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

                    </td>

                  </tr>

            </tbody><br>

        </table>

    </div>

  </div>

  <div class="modal" id="myModel">

          <div class="model-dialog">

            <div class="model-content">

                  <form [formGroup]=" userupdatefrom" #updateuser (ngSubmit)="update(user)">

              <!-- Modal Header -->

              <div class="model-header">

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

              </div>

              <!-- Modal body -->

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

                  <div [hidden]="isupdated">

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

                      <div class="from-group">

                        <label for="firstName">FirstName</label>

                        <input type="text" class="from-control" fromControlName="firstName" [(ngModel)]="user.firstName">

                    </div>

                    <div class="from-group">

                        <label for="lastName">LastName</label>

                        <input type="text" class="from-control" fromControlName="lastName" [(ngModel)]="user.lastName">

                    </div>

                    <div class="from-group">

                        <label for="userName">UserName</label>

                        <input type="text" class="from-control" fromControlName="userName" [(ngModel)]="user.userName">

                    </div>

                    <div class="from-group">

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

                        <input type="text" class="from-control"  fromControlName="password" [(ngModel)]="user.passwoed">

                </div>

                <div class="from-group">

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

                    <input type="text" class="from-control" fromControlName="email" [(ngModel)]="user.email">

                </div>

                <div class="from-group">

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

                    <select class="from-control" fromControlName="role" required>

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

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

                    </select>

                </div>

                <div class="from-group">

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

                    <input type="text" class="from-control" fromControlName="status" [(ngModel)]="user.status">

                </div>

                <div class="from-group">

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

                    <input type="text" class="from-control" fromControlName="isOnline" [(ngModel)]="user.isOnline">

                </div>

                <div class="from-group">

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

                    <input type="text" class="from-control" fromControlName="enabled" [(ngModel)]="user.enabled">

                </div>

                    </div>

                    <div [hidden]="!isupdated">

                        <h4>User Detail Updated!</h4>

                    </div>

              </div>

              <!-- Modal footer -->

              <div class="model-footer" >

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

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

              </div>

          </form>

            </div>

          </div>

        </div>

***User*-list components.css**:

.panel{

    position: relative;

    left:40px;

    background-color: rgb(89, 0, 255);

}

td{

    border-right: solid 0.4px #0a0a0a;

    border-bottom: solid 0.4px black;

}

th{

    border-top:solid o.4px black;

    border-buttom:solid o.4px black;

}

**Register-user components.ts:**

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.css']

})

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 ] ),

    role:new FormControl( )

  });

  register(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.createUsert(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');

  }

  registrationFrom(){

    this.submitted=false;

    this.registrationform.reset();

  }

  logInUser() {

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

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

  }

}

**Register-user components.html:**

<h3>Registration Here</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(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" >

              <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 lastName">

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

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

                  <span \*ngIf="LastName.errors.required">Last Name 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">

              <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="text" class="form-control" formControlName="password"

            data-toggle="tooltip" data-placement="right" title="Enter 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="text" class="form-control" formControlName="confirm\_password"

            data-toggle="tooltip" data-placement="right" title="Enter confirm\_password">

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

            style="margin-top: 5px;">

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

                <span \*ngIf = "Confirm\_Password.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">

            <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.minlength">

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

</div>

<div class="row">

  <div class="col-sm-4"></div>

  <div class="col-sm-4">

      <div [hidden]="!submitted">

         <h4>Congtatulations! you have registered successFully!</h4>

      </div>

  </div>

</div>

**Register-user components.css:**

body {

    margin: 0;

    padding: 0;

    font-family: sans-serif;

    background: -webkit-gradient( left top, right top, from(#3498db), to(#1565c0));

    background: linear-gradient(to left, #3498db, #1565c0);

  }

  .row {

    width: 1300px;

    hidth: 10px;

    position: relative;

    border: 2px solid #2ecc50;

    border-radius: 25px;

    background:rgb(68, 0, 255);

    text-align: center;

    -webkit-transition: 0.25s;

    transition: 0.25s;

    margin-top: 100px;

  }

  .row input[type="text"],

  .row input[type="password"],

  .row select {

    border: 0;

    background: none;

    display: block;

    margin: 15px auto;

    text-align: center;

    border: 1px solid #bcdb34;

    padding: 10px 10px;

    width: 250px;

    outline: none;

    color:white;

    border-radius: 24px;

    -webkit-transition: 0.25s;

    transition: 0.25s;

  }

  .row label {

    color: aliceblue;

  }

  .row h2, h4 {

    color: black;

    text-transform: uppercase;

    font-weight: 800;

  }

  .row label:hover {

    color:black;

  }

  .row h2:hover {

    color: yellow;

  }

  .row a {

    color: white;

  }

  .row a:hover {

    color: yellow;

  }

  .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;

    -webkit-transition: 0.25s;

    transition: 0.25s;

    cursor: pointer;

  }

  .row

  button[type="submit"]:hover {

    background: #2ecc71;

  }

  ::-webkit-input-placeholder {

    color: white;

    opacity: 1;

    /\* Firefox \*/

  }

  :-ms-input-placeholder {

    color: white;

    opacity: 1;

    /\* Firefox \*/

  }

  ::-ms-input-placeholder {

    color: white;

    opacity: 1;

    /\* Firefox \*/

  }

  ::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: 50%;

    visibility: hidden;

  }

  .form-control {

    width: 55%;

    border-radius: 5px;

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

    font-size: 13px;

    font-family: monospace;

  }

**Active-user components.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.css']

})

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 components.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>FirstName</th>

                        <th>LastName</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>

**Active-user components.css:**

.panel {

    position: relative;

    left: 40px;

    background-color: rgb(76, 0, 255);

  }

  td {

    border-right: solid 0.3px #0a0a0a;

    border-bottom: solid 0.3px black;

  }

  th {

    border-top: solid 0.5px black;

    border-bottom: solid 0.5px black;

  }

**Login-user components.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.css']

})

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 components.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 components.css:**

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%;

    }

  }

**Blog-list components.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.css']

})

export class BlogListComponent implements OnInit {

  constructor(private blogservice: BlogService) { }

  blog: Blog =new Blog();

  submitted = false;

  ngOnInit(): void {

    this.submitted = false;

  }

}

**Blog-list components.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>Joyasree Mondal</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 components.css:**

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 {

    -webkit-box-shadow: none;

            box-shadow: none;

  }

  .name {

    color: #007bff;

  }

  .cursor:hover {

    color: blue;

  }

  .cursor {

    cursor: pointer;

  }

  .textarea {

    resize: none;

  }

**Home components.ts:**

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

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

@Component({

  selector: 'app-home',

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

  styleUrls: ['./home.component.css']

})

export class HomeComponent implements OnInit {

  constructor(private router : Router) { }

  ngOnInit(): void {

  }

  logInUser() {

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

  }

  RegUser(){

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

  }

}

**Home components.html:**

<div class="container">

    <div class="navbar">

        <div class="menu">

            <h1 class="logo"><span>OnlineCollaborate</span></h1>

        </div>

    </div>

    <div class="main-container">

        <div class="main">

            <header>

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

                    <h2 class="title">online collaboration</h2>

                    <div class="bts">

                        <a mat-list-item  class="nav-link" (click)="logInUser()" class="btn">Login</a>/

                        <a mat-list-item   class="nav-link" (click)="RegUser()" class="btn">Resgistration</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 components.css:**

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: 1.8rem; font-weight: 800; text-transform: uppercase;

    letter-spacing: 1px; line-height: 6rem; margin-top: 20px;

    color:white;

}

.logo span{ font-size: 4.0rem; margin-right: 20px; color:rgb(184, 11, 69); }

.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;

}

.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;

}

.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;

}

.pages{

    position:absolute;

    top:-20%;

    left:35%;

    visibility: hidden;

}

**User-profile components.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-user-profile',

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

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

})

export class UserProfileComponent implements OnInit {

  constructor(private userservice: UserService) { }

  users!: Observable <User[]>;

  user: User = new User();

  userlist: any;

  ngOnInit(): void {

  }

}

**User-profile components.html:**

<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://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.niit.com%2Findia%2F&psig=AOvVaw0j9s-Yf0kBHn8B3F0cCeOE&ust=1615370269406000&source=images&cd=vfe&ved=0CAMQjB1qFwoTCKjq9635ou8CFQAAAAAdAAAAABAD" class="img-radius" alt="User-Profile-Image"> </div>

                                    <h6 class="f-w-600">Joyasree</h6>

                                    <p>Web Designer</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">Joyasree Mondal</h6>

                                        </div>

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

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

                                            <h6 class="text-muted f-w-400">joyasree.niit.mondal@gmail.com</h6>

                                        </div>

                                    </div>

                                </div>

                            </div>

                        </div>

                    </div>

                </div>

            </div>

        </div>

    </div>

    </div>

**Nav components.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.css']

})

export class NavComponent  {

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

  .pipe(

    map(result => result.matches),

    shareReplay()

  );

  constructor(private breakpointObserver: BreakpointObserver ) {}

}

**Nav components.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-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;">collaboration</span>

      </mat-toolbar>

    </mat-sidenav-content>

  </mat-sidenav-container>

**Nav components.css:**

.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;

  }

**Navigation components.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.css']

})

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(): void {

    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 components.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;">OnlineCollaborate</span>

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

    </mat-toolbar>

  </mat-sidenav-content>

</mat-sidenav-container>

<p class="welcome"></p>

<router-outlet></router-outlet>

**Navigation components.css:**

.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 components.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-user',

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

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

})

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(): void {

    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 components.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>OnlineCollaborate</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 components.css:**

.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;

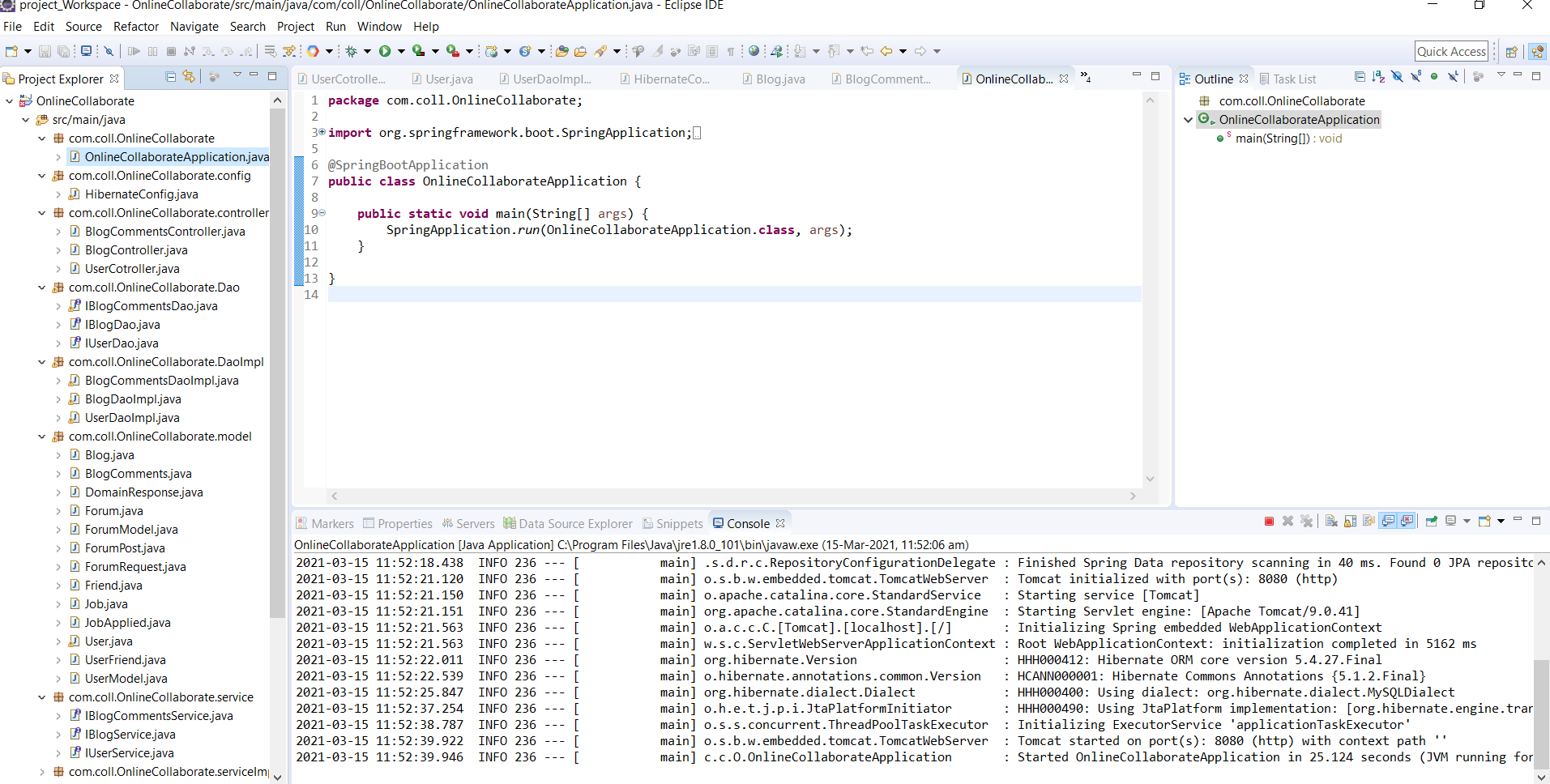
    border-radius: 8px;

    border-color:black;

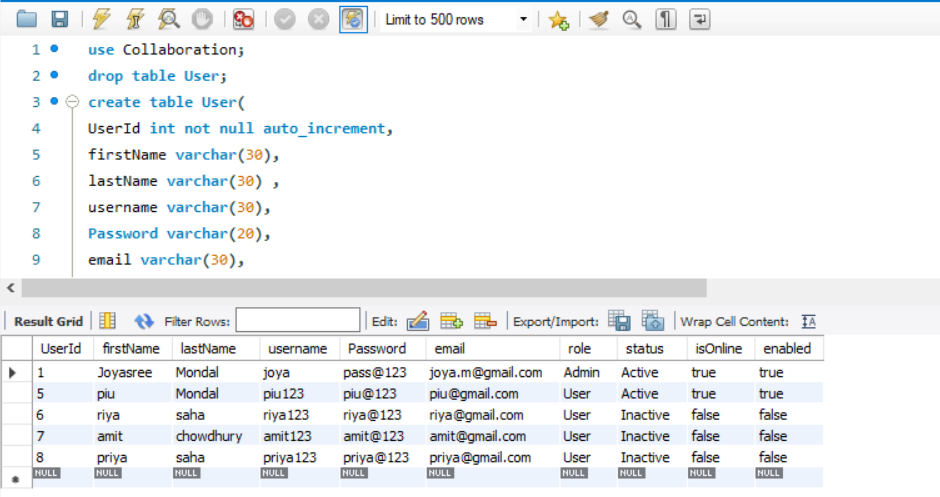
  }

* Save all
* Run the SpringBoot Application
* Run the Angular project

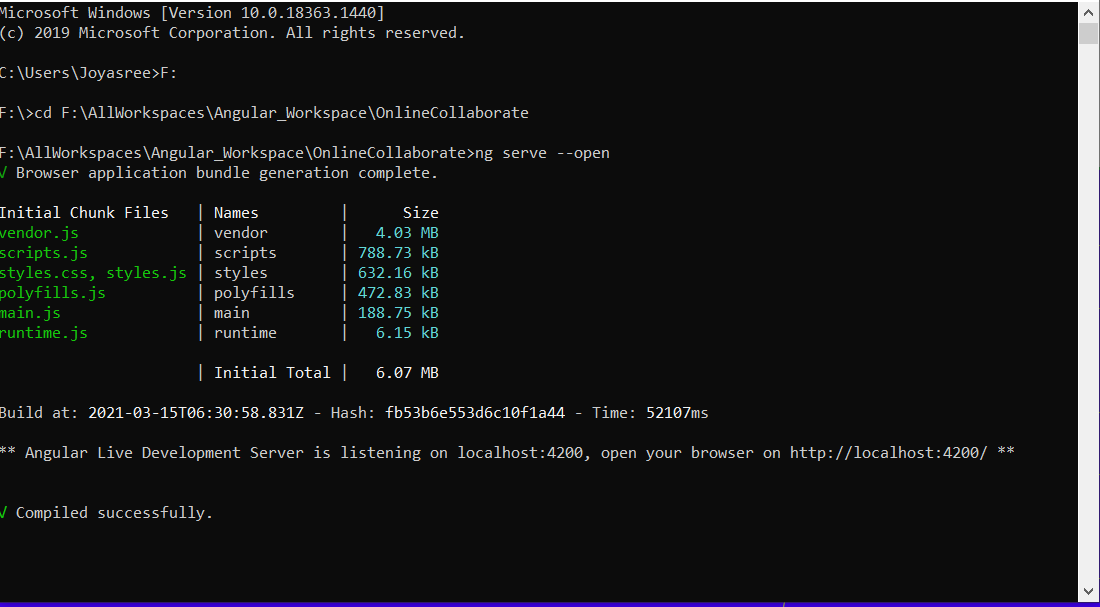
Running eclipse:



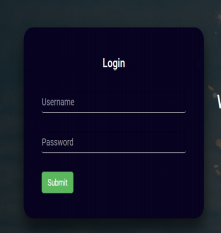
**Running Mysql:**

****

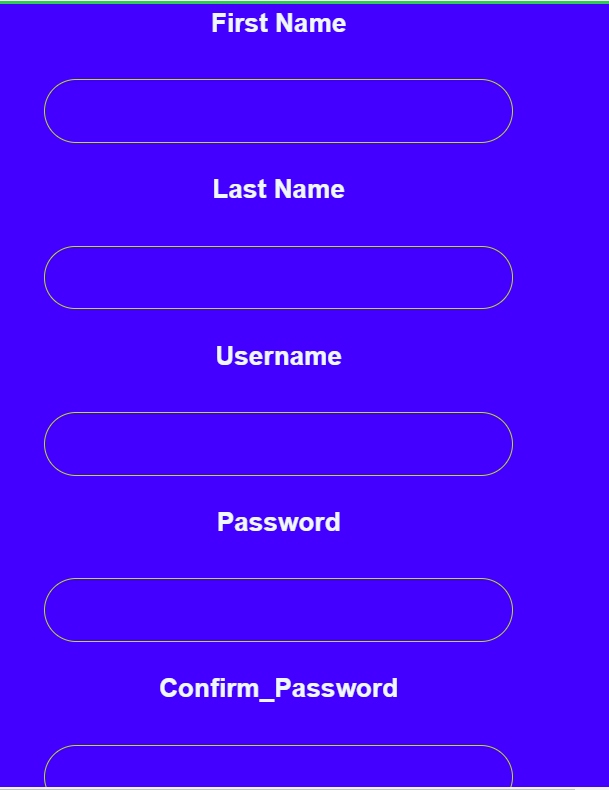
**Running Angular:**

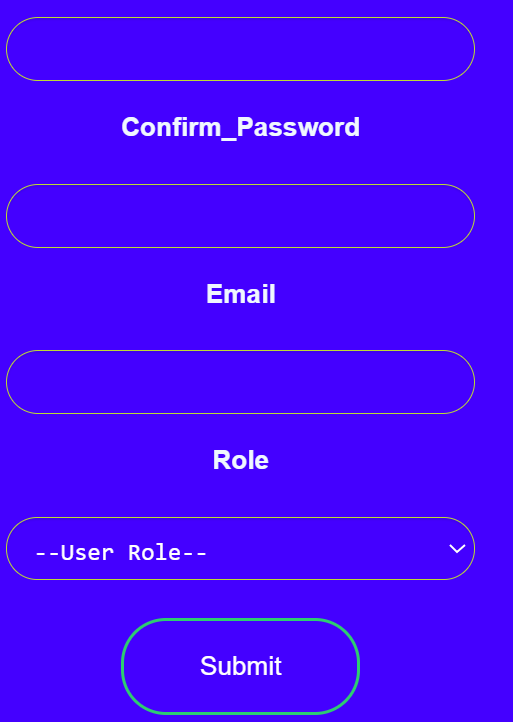
****

**Login**

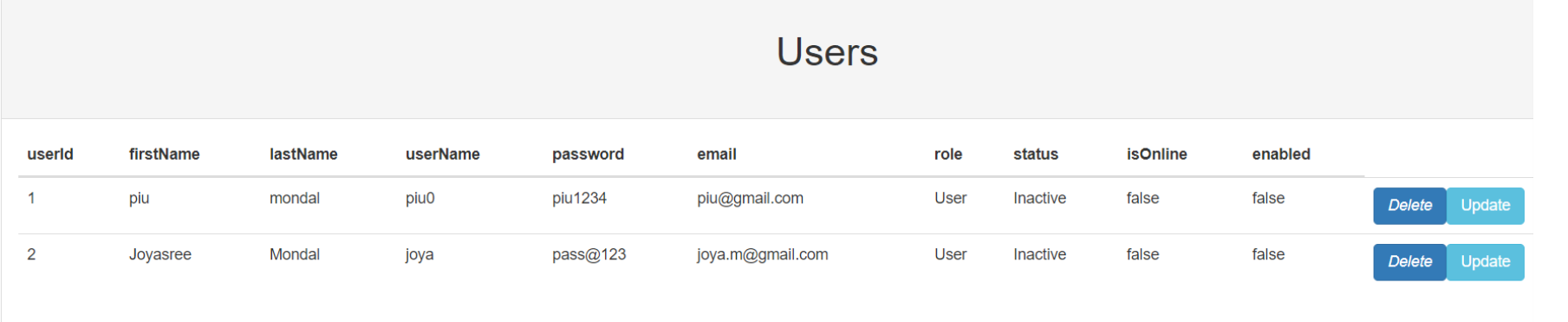
****

**Registration:**

****



**User-list:**

****