**Online Collaboration**

**Project By -** Niranjan V

**Reg No. -** S200032700155

**Online Collaboration**

Batch Code - S210167

Start Date - 05/01/21

End Date - 31/03/21

Name Of the Coordinator - Lopamudra Bera

Name Of the Developer - Niranjan V

**NIIT**

###### CERTIFICATE

This is to certify that this report, titled Online Collaborate embodies the original work done by Niranjan V, in fulfilling his course requirement at NIIT.

Coordinator - Lopamudra Bera

**Acknowledgement**

I would like to thank my mentor Lopamudra Bera and also the institute NIIT who helped me achieve this project on Online Collaborate. It helped me to do a lot on the research on the project and the environment.

I am really thankful for the staffs for helping me around in learning a lot of information during my learning process.

**Abstract**

This project works as an Platform which will be used to connect with other people in order to search a job or in order to visit their Blogs.

Admin will have the control for the application where he can delete user’s, blogs etc..

Anyone can register with our application but admin will have to approve the user in order to enter the application.

**Also, the blog control will be with the Admin, so there will be no unnecessary blogs.**

**Configuration’s**

Hardware - MacBook Pro (i9 processor) 16Gb RAM,

1TB Hard Disk.

Operating System - Mac Os Catalina X64

Software - Eclipse, MySQL, JRE, Angular JS,

Spring-boot, Hibernate

**Contents:**

Chapter 1- Introduction

1. 1.0 Aim
2. 1.1 Objectives
3. 1.2 Case Study

Chapter 2- Requirements

1. Statement of Requirements
2. Project Plan

Chapter 3- Project Design

1. User Interface Design
2. Database Structure
3. Project Life-Cycle

Chapter 4- Codes

1. Packages Created in the Backend
2. Components in Front end.

Chapter 1- Introduction

* 1. Aim -

One of the Main Objective of the project is to develop an Application that could help a lot people to connect with each other and also to post their thoughts on their profile.

* 1. Objectives -

The Objectives that are to be achieved through Online Collaboration Application are,

1. User Registration
2. Login
3. Home Page
4. About Page
5. Admin- Login
6. User – Login
7. Blog – Posting

This Web-Application can do the following above Functionalities.

This project works on the usage of all the bloggers around the globe to connect with each-other through this application which will be fully controlled by the Admin.

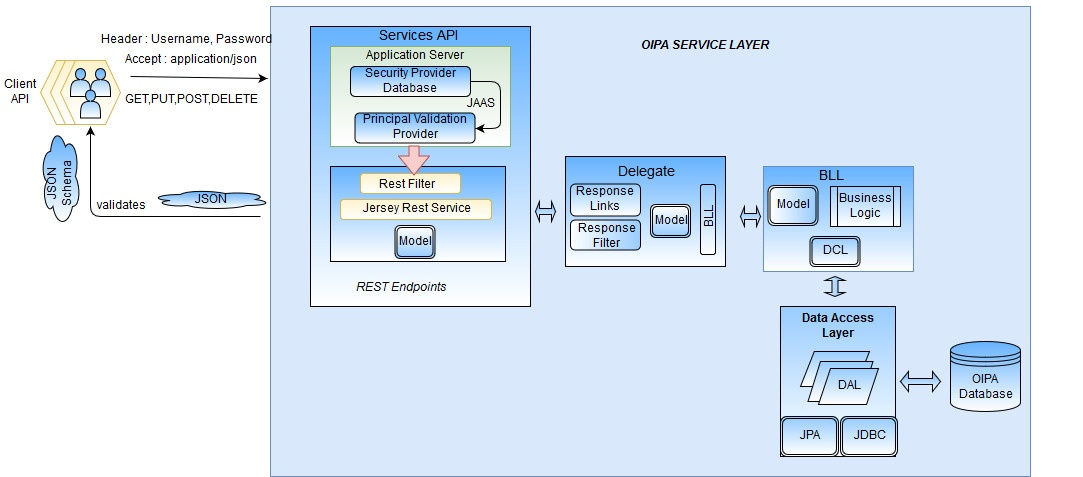
Chapter 2- Requirements

1. Statement of Requirements:
   1. ECLIPSE
   2. MYSQL
   3. VISUAL STUDIO
   4. MS WORD
   5. MS EXCEL

**PROJECT PLAN**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Project phase** | **Week 1** | **Week 2** | **Week 3** | **Week 4** | **Week 5** | **Week 6** | **Week 7** |
| **Configuration,**  **Connection** |  |  |  |  |  |  |  |
| **Backend and**  **Database** |  |  |  |  |  |  |  |
| **Frontend** |  |  |  |  |  |  |  |
| **Designing** |  |  |  |  |  |  |  |

**PROJECT LIFE CYCLE**



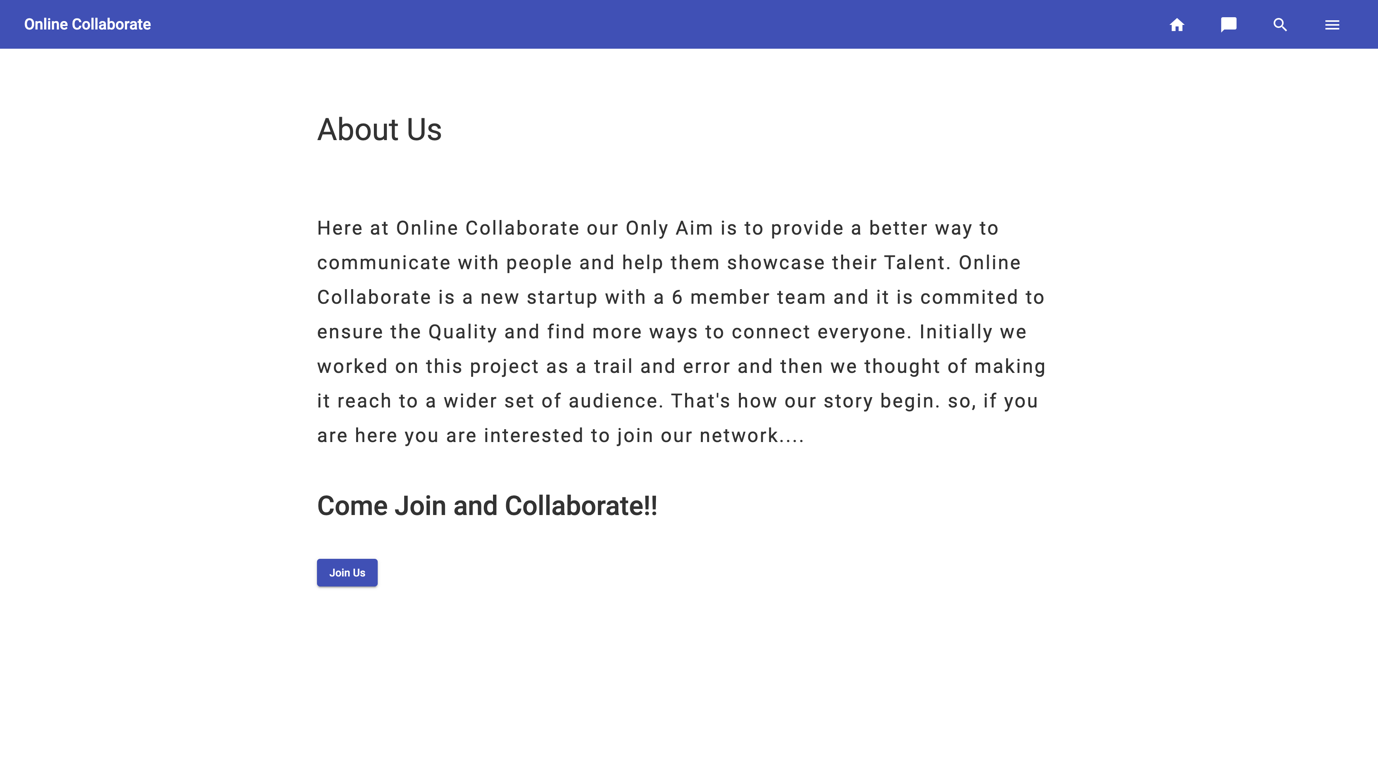
Chapter 3- Project Design

1. User Interface -

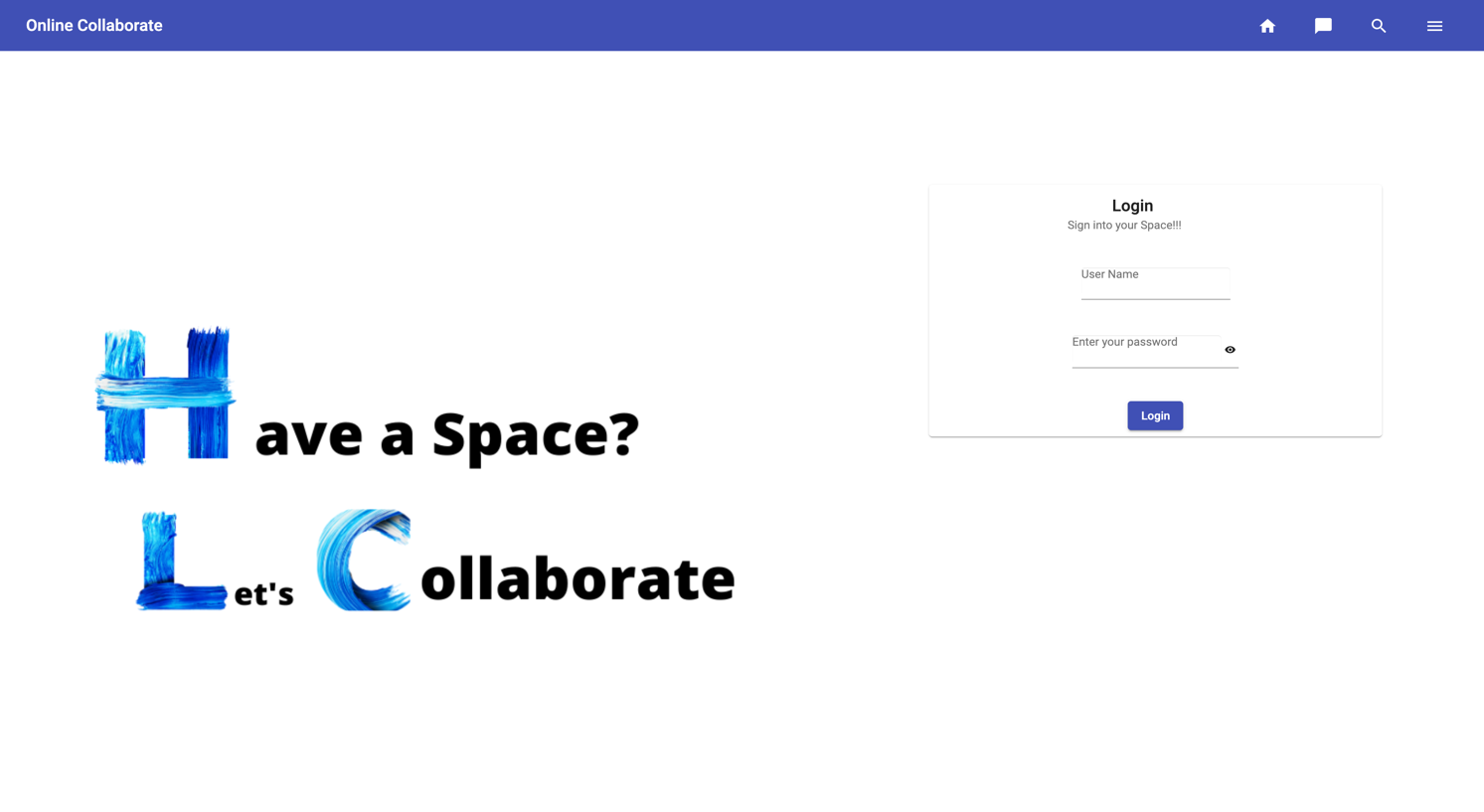
* Home Page :



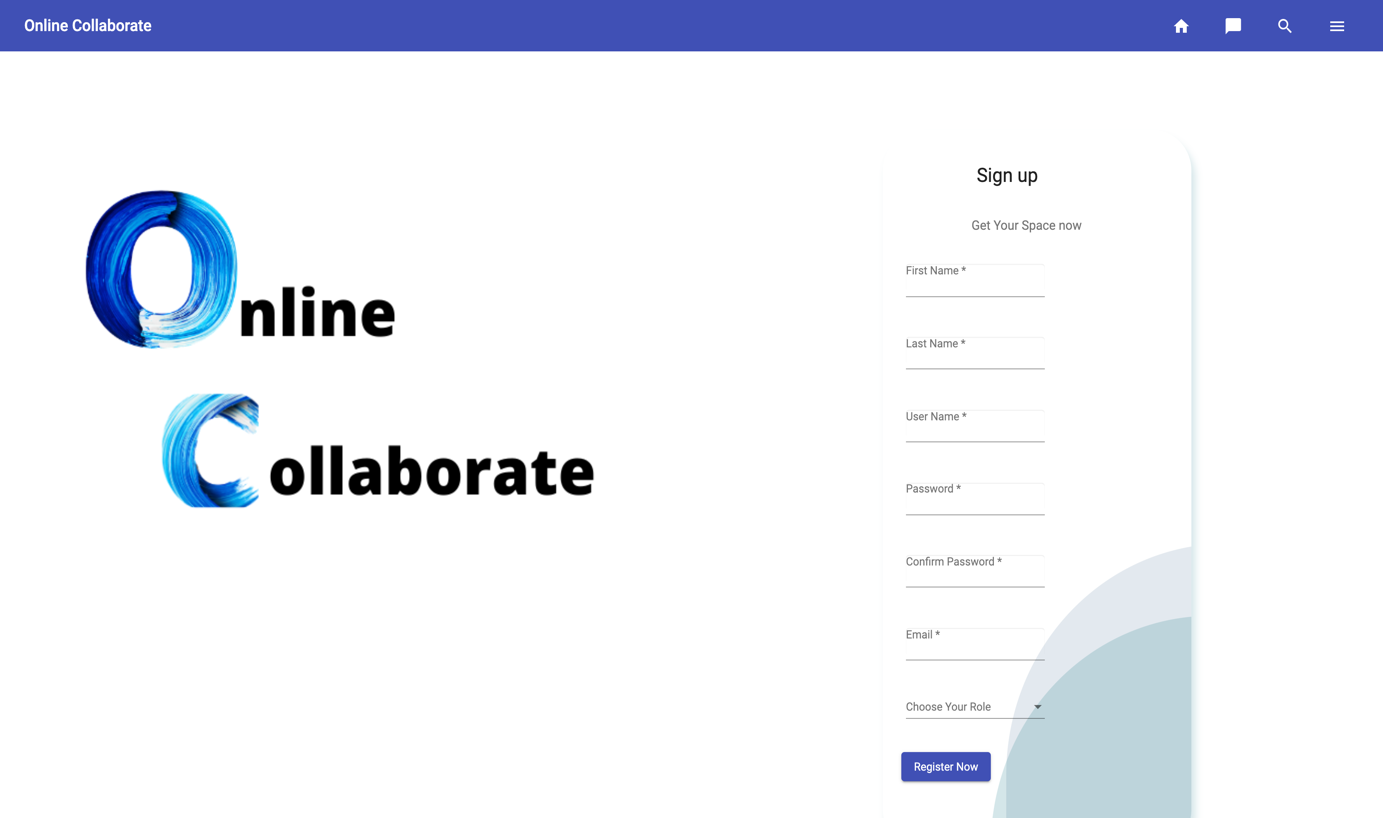
* About Page:



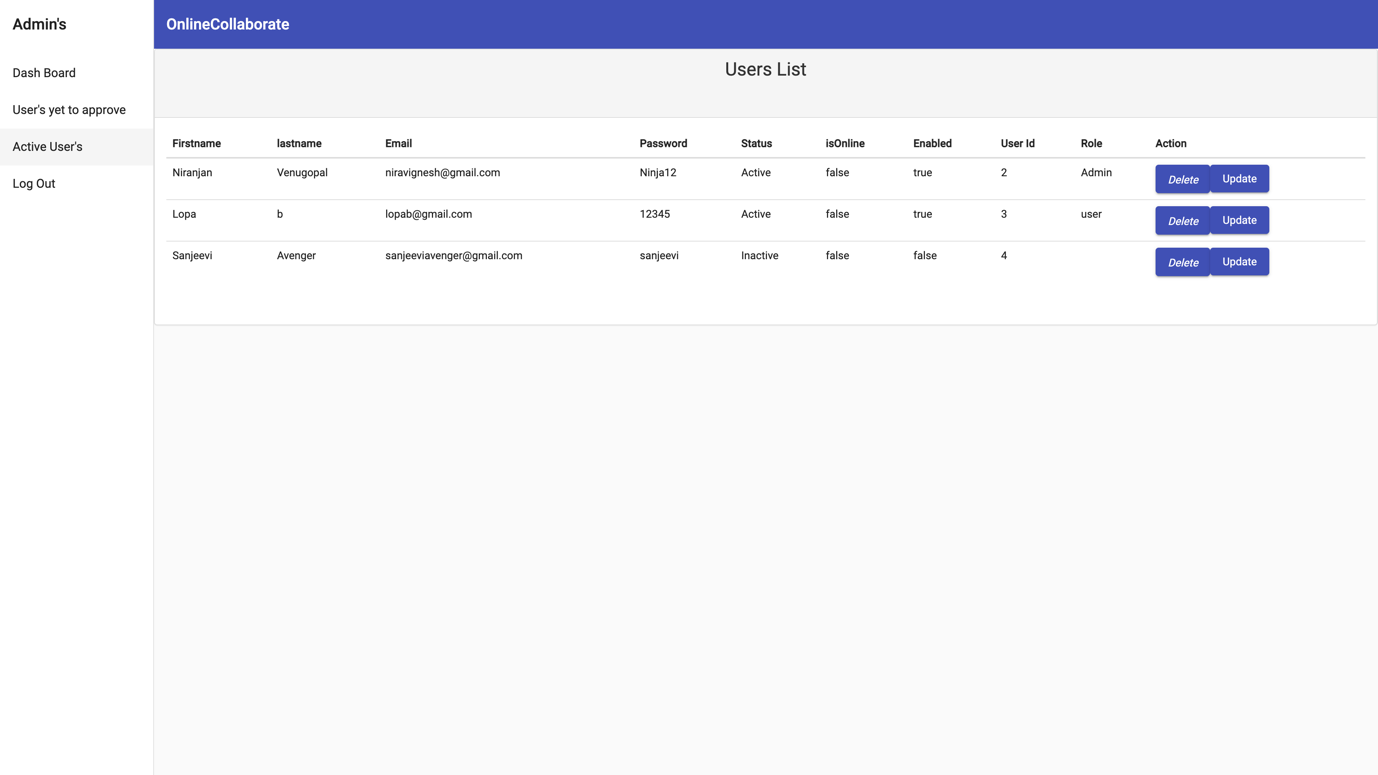
* Login Page :



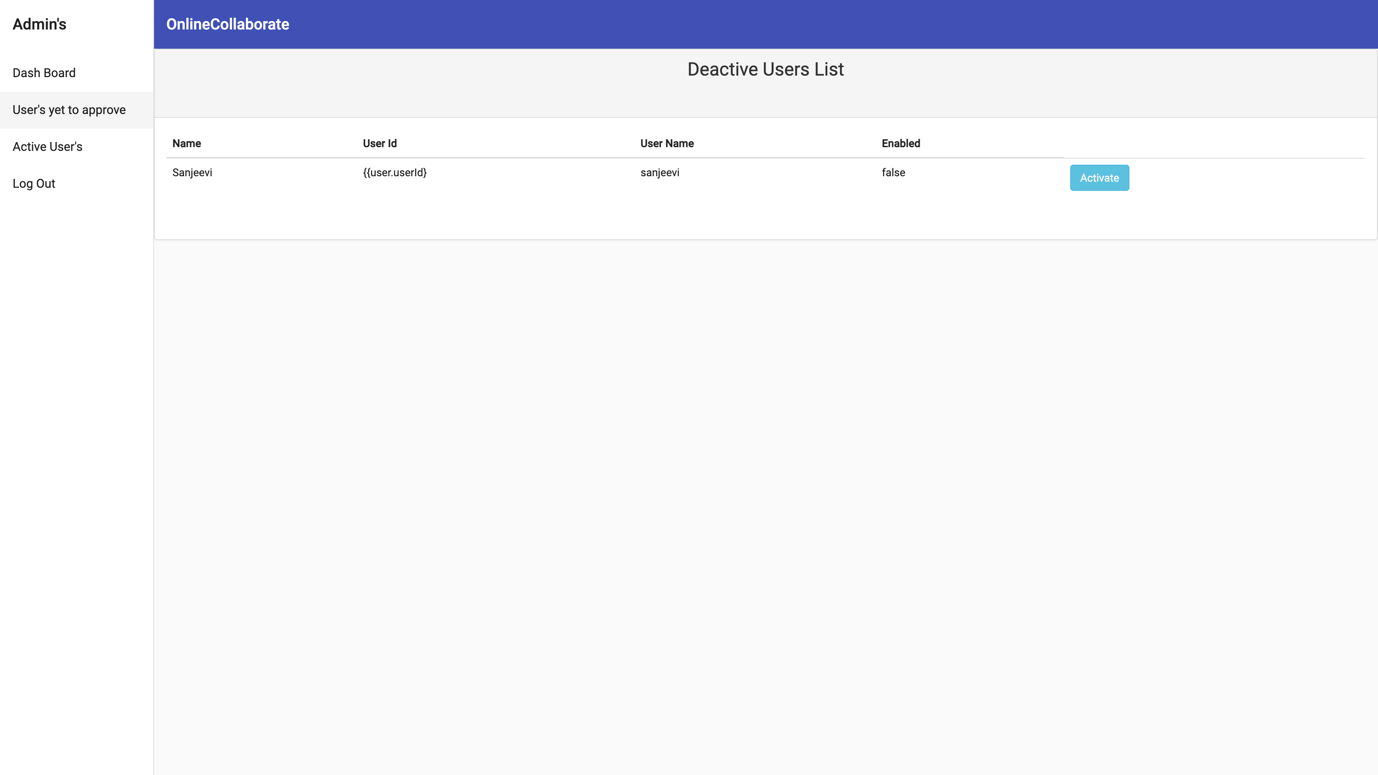
* Sign Up Page:



* Active Users inside (Admin’s Dashboard) :



* Activate users (inside Admin’s Dashboard) :



**Database Structure:**

* **User Table**

create table User(

UserId int not null auto\_increment,

FirstName varchar(30),

LastName varchar(20),

UserName varchar(20),

Password varchar(20),

Email varchar(30),

Role varchar(5),

Status varchar(10),

IsOnline varchar(5),

Enabled varchar(5),

primary key(UserId)

)

* **Blog Table**

create table Blog(

BlogId int not null auto\_increment,

BlogTitle varchar(50),

BlogContent varchar(200),

BlogPosted Date,

Status varchar(10),

NoOfLikes int,

NoOfComments int,

NoOfViews int,

UserId int,

UserName varchar(20),

primary key(BlogId)

)

* Blog Comments Table:

create table BlogComments(

BlogCommentId int not null auto\_increment,

UserId int,

UserName varchar(20),

UserProfileId varchar(20),

Title varchar(30),

NoOfLikes int,

BlogComment varchar(50),

Current Date,

BlogId int,

primary key(BlogCommentId)

)

**Packages created**

We used Java eclipse for the backend and our project is created with the help of Spring-boot. In java eclipse we installed the dependencies and we started to create packages according to our application needs. There are totally eight(8) packages created inside the “src/main/java” folder. The following packages should be created,

* com.coll.OnlineCollaborate
* com.coll.OnlineCollaborate.config
* com.coll.OnlineCollaborate.model
* com.coll.OnlineCollaborate.controller
* com.coll.OnlineCollaborate.dao
* com.coll.OnlineCollaborate.daoImpl
* com.coll.OnlineCollaborate.service
* com.coll.OnlineCollaborate.serviceImpl

After creating the packages the very first step to continue is to, start with the first package which is com.coll.OnlineCollaborate. Following codes should be inserted into the packages of this project.

**USER MODEL:**

**package** com.coll.OnlineCollaborate.model;

**import** java.io.Serializable;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.Transient;

**import** org.springframework.stereotype.Component;

@Component

@Entity

**public** **class** User **extends** DomainResponse **implements** Serializable {

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

@Id

@GeneratedValue(strategy=GenerationType.***IDENTITY***)

**private** **int** userId;

**private** String firstName;

**private** String lastName;

**private** String username;

**private** String password;

**private** String email;

**private** String role;

**private** String status;

**private** **boolean** isOnline;

**private** **boolean** 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** **boolean** getIsOnline() {

**return** isOnline;

}

**public** **void** setIsOnline(**boolean** isOnline) {

**this**.isOnline = isOnline;

}

**public** **boolean** getEnabled() {

**return** enabled;

}

**public** **void** setEnabled(**boolean** enabled) {

**this**.enabled = enabled;

}

**public** **static** **long** getSerialversionuid() {

**return** ***serialVersionUID***;

}

}

**BLOG MODEL:**

package com.coll.OnlineCollaborate.model;

import java.io.Serializable;

import java.time.LocalDate;

import java.util.List;

import javax.persistence.CascadeType;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.OneToMany;

import javax.persistence.OneToOne;

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;

}

**BLOGCOMMENTS MODEL:**

package com.coll.OnlineCollaborate.model;

import java.io.Serializable;

import java.time.LocalDate;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import org.springframework.stereotype.Component;

import com.fasterxml.jackson.annotation.JsonBackReference;

@Component

@Entity

public class BlogComments implements Serializable{

private static final long serialVersionUID = 1L;

@Id

@GeneratedValue(strategy=GenerationType.IDENTITY)

int blogCommenstId;

int userId;

String username;

String userProfileId;

String title;

int noOfLikes;

String blogComment;

LocalDate currentDate;

@ManyToOne

@JoinColumn(name="BlogId")

@JsonBackReference

Blog blog;

public int getBlogCommentsId() {

return blogCommenstId;

}

public void setBlogCommentId(int blogCommentsId) {

this.blogCommenstId = blogCommentsId;

}

public int getUserId() {

return userId;

}

public void setUserId(int userId) {

this.userId = userId;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getUserProfileId() {

return userProfileId;

}

public void setUserProfileId(String userProfileId) {

this.userProfileId = userProfileId;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public int getNoOfLikes() {

return noOfLikes;

}

public void setNoOfLikes(int noOfLikes) {

this.noOfLikes = noOfLikes;

}

public String getBlogComment() {

return blogComment;

}

public void setBlogComment(String blogComment) {

this.blogComment = blogComment;

}

public LocalDate getCurrentDate() {

return currentDate;

}

public void setCurrentDate(LocalDate currentDate) {

this.currentDate = currentDate;

}

public Blog getBlog() {

return blog;

}

public void setBlog(Blog blog) {

this.blog = blog;

}

public static long getSerialversionuid() {

return serialVersionUID;

}

}

5. Create the DAO interface implementation class

**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 updateUser(User user);

boolean deleteUser(int userId);

boolean deactiveUser(int userId);

boolean updateUserProfile(String file,Integer userId);

boolean addUser(User user);

boolean activeUser(int userId);

List<User> getAllDeactiveUser();

boolean logoutUser(int userId);

}

**IBlogDao:**

package com.coll.OnlineCollaborate.dao;

import java.util.List;

import com.coll.OnlineCollaborate.model.Blog;

public interface IBlogDao {

List<Blog> getAllBlogs();

List<Blog> getBlogsByStatus(String status);

List<Blog> getUsersBlogs(int id);

Blog getBlogsById(int blogId);

boolean addBlog(Blog blog);

boolean updateBlog(Blog blog);

boolean deleteBlog(Blog blog);

}

**IBlogCommentsDao:**

package com.coll.OnlineCollaborate.dao;

import java.util.List;

import com.coll.OnlineCollaborate.model.BlogComments;

public interface IBlogCommentsDao {

List<BlogComments> getAllBlogComments();

BlogComments getBlogCommentsById(int blogCommentsId);

boolean addBlogComments(BlogComments blogComments);

boolean updateBlogComments(BlogComments blogComments);

boolean deleteBlogComments(BlogComments blogComments);

List<BlogComments> getCommentsByBlog(int blogId);

}

1. Create the DAO interface implementation class

**UserDaoImpl:**

package com.coll.OnlineCollaborate.daoImpl;

import java.util.List;

import org.hibernate.SessionFactory;

import org.hibernate.query.Query;

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

import org.springframework.stereotype.Repository;

import org.springframework.transaction.annotation.Transactional;

import com.coll.OnlineCollaborate.dao.IUserDao;

import com.coll.OnlineCollaborate.model.User;

@Repository("userDao")

@Transactional

public class UserDaoImpl implements IUserDao{

@Autowired

SessionFactory sessionFactory;

@Override

public List<User> userListbyStatus(String status) {

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

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

return query.getResultList();

}

@Override

public List<User> getAllUsers() {

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

}

@Override

public User getUserById(int userId) {

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

}

@Override

public User getUserByUsername(String username) {

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

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

}

@SuppressWarnings("rawtypes")

@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();

return user;

}

catch(Exception e) {

e.printStackTrace();

return null;

}

}

@Override

public boolean addUser(User user) {

try {

sessionFactory.getCurrentSession().save(user);

System.out.println(user);

return true;

}

catch(Exception ex) {

ex.printStackTrace();

return false;

}

}

@Override

public boolean updateUser(User user) {

try {

sessionFactory.getCurrentSession().update(user);

return true;

}

catch(Exception ex) {

ex.printStackTrace();

return false;

}

}

@Override

public boolean deleteUser(int userId) {

try {

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

return true;

}

catch(Exception ex) {

ex.printStackTrace();

return false;

}

}

@Override

public boolean deactiveUser(int userId) {

try {

User user=getUserById(userId);

user.setEnabled(false);

user.setStatus("Inactive");

sessionFactory.getCurrentSession().update(user);

return true;

}

catch(Exception ex) {

ex.printStackTrace();

return false;

}

}

@Override

public boolean activeUser(int userId) {

try {

User user=getUserById(userId);

user.setEnabled(true);

user.setStatus("Active");

sessionFactory.getCurrentSession().update(user);

return true;

}

catch(Exception ex) {

ex.printStackTrace();

return false;

}

}

@Override

public 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 logoutUser(int userId) {

try {

User user=getUserById(userId);

user.setIsOnline(false);

sessionFactory.getCurrentSession().update(user);

return true;

}

catch(Exception ex) {

ex.printStackTrace();

return false;

}

}

}

**BlogDaoImpl:**

package com.coll.OnlineCollaborate.daoImpl;

import java.util.List;

import 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.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 id) {

String q="select blogTitle,blogContent from Blog where blogId='"+id+"'";

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

return query.getResultList();

}

@Override

public Blog getBlogsById(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;

}

}

**BlogCommentsDaoImpl:**

package com.coll.OnlineCollaborate.daoImpl;

import java.util.List;

import org.hibernate.SessionFactory;

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

import org.springframework.stereotype.Repository;

import org.springframework.transaction.annotation.Transactional;

import com.coll.OnlineCollaborate.dao.IBlogCommentsDao;

import com.coll.OnlineCollaborate.model.Blog;

import com.coll.OnlineCollaborate.model.BlogComments;

@Repository("blogCommentsDao")

@Transactional

public class BlogCommentsDaoImpl implements IBlogCommentsDao {

@Autowired

SessionFactory sessionFactory;

@Override

public List<BlogComments> getAllBlogComments() {

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

}

@Override

public BlogComments getBlogCommentsById(int 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().saveOrUpdate(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;

}

}

@Override

public List<BlogComments> getCommentsByBlog(int blogId) {

// TODO Auto-generated method stub

return null;

}

}

7Create the service layer interface

**IUserService:**

package com.coll.OnlineCollaborate.service;

import java.util.List;

import com.coll.OnlineCollaborate.model.User;

public interface IUserService {

List<User> userListbyStatus(String status);

List<User> getAllUsers();

User getUserById(int userId);

User getUserByUsername(String username);

User validateUser(User user);

boolean updateUser(User user);

boolean deleteUser(int userId);

boolean deactiveUser(int userId);

boolean updateUserProfile(String file,Integer userId);

boolean addUser(User user);

List<User> getAllDeactiveUser();

boolean activeUser(int userId);

boolean logoutUser(int userId);

}

**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 id);

Blog getBlogsById(int blogId);

boolean addBlog(Blog blog);

boolean updateBlog(Blog blog);

boolean deleteBlog(Blog blog);

}

**IBlogCommentsService:**

package com.coll.OnlineCollaborate.service;

import java.util.List;

import com.coll.OnlineCollaborate.model.BlogComments;

public interface IBlogCommentsService {

List<BlogComments> getAllBlogComments();

BlogComments getBlogCommentsById(int blogCommentsId);

boolean addBlogComments(BlogComments blogComments);

boolean updateBlogComments(BlogComments blogComments);

boolean deleteBlogComments(BlogComments blogComments);

List<BlogComments>getCommentsByBlog(int blogId);

}

8. Create the service layer implementation class

**UserServiceImpl:**

package com.coll.OnlineCollaborate.serviceImpl;

import java.util.List;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.coll.OnlineCollaborate.dao.IUserDao;

import com.coll.OnlineCollaborate.model.User;

import com.coll.OnlineCollaborate.service.IUserService;

@Service

@Transactional

public class UserServiceImpl implements IUserService{

@Autowired

IUserDao userDao;

@Override

public List<User> userListbyStatus(String status) {

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 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 boolean addUser(User user) {

return userDao.addUser(user);

}

@Override

public List<User> getAllDeactiveUser() {

return userDao.getAllDeactiveUser();

}

@Override

public boolean activeUser(int userId) {

return userDao.activeUser(userId);

}

@Override

public boolean logoutUser(int userId) {

// TODO Auto-generated method stub

return userDao.logoutUser(userId);

}

}

**BlogServiceImpl:**

package com.coll.OnlineCollaborate.serviceImpl;

import java.util.List;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.coll.OnlineCollaborate.dao.IBlogDao;

import com.coll.OnlineCollaborate.model.Blog;

import com.coll.OnlineCollaborate.service.IBlogService;

@Service

@Transactional

public class BlogServiceImpl implements IBlogService {

@Autowired

IBlogDao blogDao;

@Override

public List<Blog> getAllBlogs() {

return blogDao.getAllBlogs();

}

@Override

public List<Blog> getBlogsByStatus(String status) {

return blogDao.getBlogsByStatus(status);

}

@Override

public List<Blog> getUsersBlogs(int id) {

return blogDao.getUsersBlogs(id);

}

@Override

public Blog getBlogsById(int blogId) {

return blogDao.getBlogsById(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);

}

}

**BlogCommentsServiceImpl:**

package com.coll.OnlineCollaborate.serviceImpl;

import java.util.List;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.coll.OnlineCollaborate.dao.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 id) {

return blogDao.getUsersBlogs(id);

}

@Override

public Blog getBlogsById(int blogId) {

return blogDao.getBlogsById(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);

}

}

9.Create the controller class

**UserController:**

package com.coll.OnlineCollaborate.controller;

import java.util.List;

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

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

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

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

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

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

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

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

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

import com.coll.OnlineCollaborate.model.User;

import com.coll.OnlineCollaborate.service.IUserService;

@RestController

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

@RequestMapping(value="/api")

public class UserController {

@Autowired

IUserService userService;

@PostMapping("save-user")

public boolean saveUser(@RequestBody User user) {

return userService.addUser(user);

}

@GetMapping("user-list")

public List<User> allUsers(){

return userService.getAllUsers();

}

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

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

return userService.deleteUser(userId);

}

@GetMapping("user/{userId}")

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

return userService.getUserById(userId);

}

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

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

return userService.updateUser(user);

}

@GetMapping("deactive-list")

public List<User> AllDeactiveUser(){

return userService.getAllDeactiveUser();

}

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

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

return userService.activeUser(userId);

}

@PostMapping("validate-user")

public User validateUser(@RequestBody User user) {

return userService.validateUser(user);

}

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

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

return userService.logoutUser(userId);

}

}

**BlogController:**

package com.coll.OnlineCollaborate.controller;

import java.util.List;

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

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

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

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

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

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

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

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

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

import com.coll.OnlineCollaborate.model.Blog;

import com.coll.OnlineCollaborate.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);

}

@PostMapping("blog-list")

public List<Blog> allBlogs(){

return blogService.getAllBlogs();

}

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

public boolean deleteUser(@PathVariable("blog") Blog blog) {

return blogService.deleteBlog(blog);

}

@GetMapping("blog/{blogId}")

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

return blogService.getBlogsById(blogId);

}

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

public boolean updateBlog(@RequestBody Blog blog,@PathVariable("blogId") int blogId) {

blog.setBlogId(blogId);

return blogService.updateBlog(blog);

}

**BlogCommentsController:**

package com.coll.OnlineCollaborate.controller;

import java.util.List;

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

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

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

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

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

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

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

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

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

import com.coll.OnlineCollaborate.model.BlogComments;

import com.coll.OnlineCollaborate.service.IBlogCommentsService;

@RestController

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

@RequestMapping(value="/api")

public class BlogCommentsController {

@Autowired

IBlogCommentsService blogCommentsService;

@PostMapping("save-blogComments")

public boolean saveBlogComments(@RequestBody BlogComments blogComments) {

return blogCommentsService.addBlogComments(blogComments);

}

@GetMapping("blogComments-list")

public List<BlogComments> allBlogComments(){

return blogCommentsService.getAllBlogComments();

}

@DeleteMapping("delete-blogComments/{blogCommentsId")

public boolean deleteBlogComments(@PathVariable("blogCommentsId") BlogComments blogCommentsId) {

return blogCommentsService.deleteBlogComments(blogCommentsId);

}

@GetMapping("blogComments/{blogCommentsId}")

public BlogComments BlogCommentsById(@PathVariable("blogCommentsId")int blogCommentsId) {

return blogCommentsService.getBlogCommentsById(blogCommentsId);

}

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

public boolean updateBlogComents(@RequestBody BlogComments blogComments,@PathVariable("blogCommentsId")int blogCommentsId) {

blogComments.setBlogCommentId(blogCommentsId);

return blogCommentsService.updateBlogComments(blogComments);

}

}

**Step – by -Step process of Angular Application:**

1. .Open Visual Studio Code. Create new project inside Angular\_Workspace. Set the project name as OnlineCollaborateAngular
2. ng new OnlineCollaborateAngular
3. Install Bootstrap CSS framework
4. Use the following command to install bootstrap in the project.
5. *D:\All\_Workspace\Angular\_Workspace\SrudentAngular>****npm install*** [***bootstrap@3.3.7***](mailto:bootstrap@3.3.7) ***–save***
6. Now, include the following code in the style.css file.
7. @import "~bootstrap/dist/css/bootstrap.css";
8. Install Angular Data Table
9. Use the following command to install bootstrap in the project.
10. *PS D:\All\_Workspace\Angular\_Workspace\OnlineCollaborateAngular>****npm install angular-datatable –save***
11. It is required to include **DataTableModule** in imports array of **app.module.ts** file
12. Generate Components  
    Open the project in visual studio and then use the following command to generate Angular components:
13. ng g c RegisterUser
14. ng g c UserList
15. ng g c LoginUser
16. ng g c ActiveUser
17. ng g c BlogList
18. ng g c Home
19. ng g c UserProfile
20. ng g c Navigate
21. ng g c NavigateAdmin
22. ng g c NavigateUser
23. Let's also create a service class by using the following command: -
24. ng g s user
25. ng g s blog
26. Edit the **app.module.ts** file
    1. **Import Routing** - Here, we are importing routing file (app-routing.module.ts) and include it in imports array.
    2. **Import ReactiveFormsModule** - Here, we are importing **ReactiveFormsModule** for reactive forms and specify it in imports array.
    3. **Import HttpModule** - Here, we are importing **HttpModule** for server requests and specifying it in imports array.
    4. **Register Service class** - Here, we are mentioning the service class in provider's array.

//app.module.ts

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

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

import { CommonModule } from "@angular/common";

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

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

import {DataTablesModule} from 'angular-datatables';

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

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

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

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 { RouterLinkWithHref, RouterModule } from '@angular/router';

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

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

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

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

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

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

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

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

@NgModule({

  declarations: [

    AppComponent,

    RegisterUserComponent,

    UserListComponent,

    LoginUserComponent,

    ActiveUserComponent,

    BlogListComponent,

    HomeComponent,

    NavComponent,

    NavigationComponent,

    NavigationUserComponent,

    UserProfileComponent

  ],

  imports: [

    BrowserModule,

    AppRoutingModule,

    RouterModule,

    DataTablesModule,

    FormsModule,

    ReactiveFormsModule,

    HttpClientModule,

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

  ],

  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }

18. Edit the **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 {BlogListComponent } from './blog-list/blog-list.component';

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

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

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

import {UserProfileComponent} from './user-profile/user-profile.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: 'register-user', component: RegisterUserComponent },

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

  ];

@NgModule({

  imports: [RouterModule.forRoot(routes)],

  exports: [RouterModule]

})

export class AppRoutingModule { }

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

<router-outlet></router-outlet>

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

ng g class User

ng g class Blog

**Implementation process of Angular Codes:**

* User.ts :
* export class User {
* userId:number;
* firstName:String;
* lastName:String;
* userName:String;
* password:String;
* confirmPassword:String;
* email:String;
* role:String;
* status:String;
* isOnline:String;
* enabled:String;
* }
* Register.ts :
* import { Component, OnInit } from '@angular/core';
* import { UserService } from '../user.service';
* import {FormControl,FormGroup,Validators} from '@angular/forms';
* import { User} from '../user';
* import { error } from '@angular/compiler/src/util';
* @Component({
* selector: 'app-register-user',
* templateUrl: './register-user.component.html',
* styleUrls: ['./register-user.component.scss']
* })
* export class RegisterUserComponent implements OnInit {
* constructor(private userservice:UserService) { }
* user : User=new User();
* submitted = false;
* ngOnInit(): void {
* this.submitted=false;
* }
* registrationform=new FormGroup({
* firstName:new FormControl('' , [Validators.required] ),
* lastName:new FormControl('' , [Validators.required] ),
* userName:new FormControl('' , [Validators.required] ),
* password:new FormControl('' , [Validators.required] ),
* confirmPassword:new FormControl('' , [Validators.required]),
* email:new FormControl('',[Validators.required,Validators.email]),
* role:new FormControl()
* });
* registerUser(registerUser){
* this.user=new User();
* this.user.firstName=this.UserFirstname.value;
* this.user.lastName=this.UserLastname.value;
* this.user.userName=this.UserName.value;
* if(this.UserPassword.value===this.UserConfirmPassword.value)
* this.user.password=this.UserPassword.value;
* this.user.email=this.UserEmail.value;
* this.user.role=this.UserRole.value;
* if(this.user.role==="Admin"){
* this.user.enabled="true";
* this.user.status="Active"
* }
* else{
* this.user.enabled="false";
* this.user.status="Inactive"
* }
* this.user.isOnline="false";
* this.submitted=true;
* console.log(this.user.firstName);
* this.save();
* }
* save(){
* this.userservice.createUser(this.user)
* .subscribe(data=> console.log(data), error => console.log(error));
* this.user= new User();
* }
* get UserFirstname(){
* return this.registrationform.get('firstName');
* }
* get UserLastname(){
* return this.registrationform.get('lastName');
* }
* get UserName(){
* return this.registrationform.get('userName');
* }
* get UserPassword(){
* return this.registrationform.get('password');
* }
* get UserConfirmPassword(){
* return this.registrationform.get('confirmPassword');
* }
* get UserEmail(){
* return this.registrationform.get('email')
* }
* get UserRole(){
* return this.registrationform.get('role');
* }
* addRegistrationForm(){
* this.submitted=false;
* this.registrationform.reset();
* }
* }
* Login.ts :
* import { Component, OnInit } from '@angular/core';
* import { Router } from '@angular/router';
* import {FormGroup, FormControl, Validators, NgForm} from '@angular/forms';
* import { User } from '../user';
* import { UserService } from '../user.service';
* import { error } from 'selenium-webdriver';
* import { DataTablesModule } from 'angular-datatables';
* import { map, shareReplay } from 'rxjs/operators';
* @Component({
* selector: 'app-login-user',
* templateUrl: './login-user.component.html',
* styleUrls: ['./login-user.component.scss']
* })
* export class LoginUserComponent implements OnInit {
* hide:any;
* 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]),
* })
* /\* submitForm(){
* let formValue = this.loginform.value;
* let userName = formValue.userName;
* let password = formValue.password;
* let blog = formValue.blog
* console.log(this.loginform.value);
* var reqBody = {
* userName : userName,
* password: password
* }
* this.\_userService(reqBody).subscribe((result) =>{
* },
* (error) => {
* })
* } \*/
* 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("/admin-nav/id");
* }
* else{
* this.router.navigateByUrl("/user-nav/id"+`${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');
* }
* }
* User-List:
* import { Component, OnInit } from '@angular/core';
* import { FormControl, FormGroup} from '@angular/forms';
* import { Observable, Subject } from 'rxjs';
* import { User } from '../user';
* import { UserService } from '../user.service';
* import {DataTablesModule} from 'angular-datatables';
* import { error } from 'protractor';
* @Component({
* selector: 'app-user-list',
* templateUrl: './user-list.component.html',
* styleUrls: ['./user-list.component.scss']
* })
* export class UserListComponent implements OnInit {
* constructor(private userservice:UserService) { }
* userArray: any[] = [];
* dtOptions: DataTables.Settings = {};
* dtTrigger: Subject<any> = new Subject();
* users: Observable <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(userId: number) {
* this.userservice.deleteUser(userId)
* .subscribe(
* data =>{
* console.log(data);
* this.deleteMessage=true;
* this.userservice.getUserList().subscribe(data =>{
* this.users = data
* })
* },
* error => console.log(error));
* }
* updateUser(userId: number){
* this.userservice.getUser(userId)
* .subscribe(
* data => {
* this.userlist=data
* },
* error => console.log(error));
* }
* userupdateform=new FormGroup({
* firstName:new FormControl(),
* lastName:new FormControl(),
* email:new FormControl(),
* role:new FormControl(),
* password:new FormControl(),
* status:new FormControl(),
* isOnline:new FormControl(),
* enabled:new FormControl()
* });
* updateUsers(updateUsers){
* this.user=new User();
* this.user.userId=this.UserId.value;
* this.user.firstName=this.UserFirstName.value;
* this.user.lastName=this.UserLastName.value;
* this.user.email=this.UserEmail.value;
* this.user.role=this.UserRole.value;
* this.user.password=this.UserPassword.value;
* this.user.status=this.UserStatus.value;
* this.user.isOnline=this.UserOnline.value;
* this.user.enabled=this.UserEnabled.value;
* console.log(this.UserFirstName.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 UserId(){
* return this.userupdateform.get('userId');
* }
* get UserFirstName(){
* return this.userupdateform.get('firstName');
* }
* get UserLastName(){
* return this.userupdateform.get('lastName');
* }
* get UserEmail(){
* return this.userupdateform.get('email');
* }
* get UserRole(){
* return this.userupdateform.get('role');
* }
* get UserPassword(){
* return this.userupdateform.get('password');
* }
* get UserStatus(){
* return this.userupdateform.get('status');
* }
* get UserOnline(){
* return this.userupdateform.get('isOnline');
* }
* get UserEnabled(){
* return this.userupdateform.get('enabled');
* }
* changeisUpdated(){
* this.isupdated=false;
* }
* }
* Activate -user:
* 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 { error } from 'selenium-webdriver';
* @Component({
* selector: 'app-activate-users',
* templateUrl: './activate-users.component.html',
* styleUrls: ['./activate-users.component.scss']
* })
* export class ActivateUsersComponent implements OnInit {
* constructor(private userservice: UserService) { }
* dtOptions: DataTables.Settings={};
* dtTrigger: Subject<any>= new Subject();
* 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.activateUser(id).
* subscribe(
* (data) =>{
* this.deactiveList =data;
* console.log(this.deactiveList);
* this.activateUser(id);
* },
* (error) => console.log(error)
* );
* }
* activeUserForm = new FormGroup({
* userId: new FormControl()
* });
* activateUser(id:number){
* this.user =new User();
* this.user.userId=id;
* this.userservice.activateUser(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');
* }
* }
* User-navigation :
* import { LayoutModule } from '@angular/cdk/layout';
* import { waitForAsync, ComponentFixture, TestBed } from '@angular/core/testing';
* import { NoopAnimationsModule } from '@angular/platform-browser/animations';
* import { MatButtonModule } from '@angular/material/button';
* import { MatIconModule } from '@angular/material/icon';
* import { MatListModule } from '@angular/material/list';
* import { MatSidenavModule } from '@angular/material/sidenav';
* import { MatToolbarModule } from '@angular/material/toolbar';
* import { UserNavComponent } from './user-nav.component';
* describe('UserNavComponent', () => {
* let component: UserNavComponent;
* let fixture: ComponentFixture<UserNavComponent>;
* beforeEach(waitForAsync(() => {
* TestBed.configureTestingModule({
* declarations: [UserNavComponent],
* imports: [
* NoopAnimationsModule,
* LayoutModule,
* MatButtonModule,
* MatIconModule,
* MatListModule,
* MatSidenavModule,
* MatToolbarModule,
* ]
* }).compileComponents();
* }));
* beforeEach(() => {
* fixture = TestBed.createComponent(UserNavComponent);
* component = fixture.componentInstance;
* fixture.detectChanges();
* });
* it('should compile', () => {
* expect(component).toBeTruthy();
* });
* });
* User-Dash:
* import { Component } from '@angular/core';
* import { map } from 'rxjs/operators';
* import { Breakpoints, BreakpointObserver } from '@angular/cdk/layout';
* import{User} from '../user';
* import { ActivatedRoute, Params } from '@angular/router';
* import { UserService } from '../user.service';
* @Component({
* selector: 'app-user-dash',
* templateUrl: './user-dash.component.html',
* styleUrls: ['./user-dash.component.scss']
* })
* export class UserDashComponent {
* /\*\* Based on the screen size, switch from standard to one column per row \*/
* cards = this.breakpointObserver.observe(Breakpoints.Handset).pipe(
* map(({ matches }) => {
* if (matches) {
* return [
* { title: 'Card 1', cols: 1, rows: 1 },
* { title: 'Card 2', cols: 1, rows: 1 },
* { title: 'Card 3', cols: 1, rows: 1 },
* { title: 'Card 4', cols: 1, rows: 1 }
* ];
* }
* return [
* { title: 'Card 1', cols: 2, rows: 1 },
* { title: 'Card 2', cols: 1, rows: 1 },
* { title: 'Card 3', cols: 1, rows: 2 },
* { title: 'Card 4', cols: 1, rows: 1 }
* ];
* })
* );
* user: any;
* id:number;
* constructor(private breakpointObserver: BreakpointObserver, private route: ActivatedRoute, private userService: UserService) {}
* ngOnInit(){
* this.route.params.subscribe (
* (params:Params)=> {
* this.id=+params["id"];
* console.log(this.id);
* this.userService.getUser(this.id).subscribe (
* data=>{
* this.user=data;
* console.log(this.user);
* }
* )
* }
* )
* }
* }
* HTML For Components:

Activate users:

<div class="panel panel-default">

<div class="panel-heading">

<h1 style="text-align: center">Deactive Users List</h1><br>

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

<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>Activated</strong>

</div>

</div>

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

</div>

</div>

<div class="panel-body">

<table class="table table-hover table-sm" >

<thead class="thead-light">

<tr>

<th>Name</th>

<th>User Id</th>

<th>User Name</th>

<th>Enabled</th>

</tr>

</thead>

<tbody>

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

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

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

</table>

</div>

</div>

User-List:

<div class="panel panel-default">

<div class="panel-heading">

<h1 style="text-align: center">Users List</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" >

<thead class="thead-light">

<tr>

<th>Firstname</th>

<th>lastname</th>

<th>Email</th>

<th>Password</th>

<th>Status</th>

<th>isOnline</th>

<th>Enabled</th>

<th>User Id</th>

<th>Role</th>

<th>Action</th>

</tr>

</thead>

<tbody>

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

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

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

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

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

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

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

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

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

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

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

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

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

</td>

</tr>

</tbody><br>

</table>

</div>

</div>

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

<div class="modal-dialog">

<div class="modal-content">

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

<!-- Modal Header -->

<div class="modal-header">

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

</div>

<!-- Modal body -->

<div class="modal-body" \*ngFor="let user of users" >

<div [hidden]="isupdated">

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

<div class="form-group">

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

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

</div>

<div class="form-group">

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

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

</div>

<div class="form-group">

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

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

</div>

<div class="form-group">

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

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

</div>

<div class="form-group">

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

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

</div>

</div>

<div [hidden]="!isupdated">

<h4>User Detail Updated!</h4>

</div>

</div>

<!-- Modal footer -->

<div class="modal-footer" >

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

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

</div>

</form>

</div>

</div>

</div>

Register-User:

<mat-toolbar color="primary" class="container-fluid">

<mat-toolbar-row>

<span>Online Collaborate</span>

<span class="example-spacer"></span>

<section>

<div class="example-button-row">

<div class="example-flex-container">

<button mat-icon-button aria-label="Home page" routerLink="/home">

<mat-icon>home</mat-icon>

</button>

<button mat-icon-button aria-label="Chat" routerLink="/about">

<mat-icon>chat\_bubble</mat-icon>

</button>

<button mat-icon-button aria-label="Search">

<mat-icon>search</mat-icon>

</button>

<button mat-icon-button [matMenuTriggerFor]="menu" class="example-icon" aria-label="Menu">

<mat-icon>menu</mat-icon>

</button>

<mat-menu #menu="matMenu">

<button mat-menu-item routerLink="/login-user">

<mat-icon>login</mat-icon>

<span>Login</span>

</button>

<button mat-menu-item routerLink="/add-user">

<mat-icon>app\_registration</mat-icon>

<span>Sign up</span>

</button>

</mat-menu>

</div>

</div>

</section>

<mat-divider></mat-divider>

</mat-toolbar-row>

</mat-toolbar>

<div class="bgimage">

<mat-card class="login">

<mat-card-title style="margin-left: 27%;">

<h1>Sign up</h1>

</mat-card-title>

<br>

<mat-card-subtitle class="mat-card-subtitle">

<h3>Get Your Space now</h3>

</mat-card-subtitle>

<form class="login-flex-container" [formGroup]="registrationform" #saveUser (ngSubmit)="registerUser(registerUser)">

<mat-card-content>

<div class="form-group">

<mat-form-field>

<mat-label for="userName">First Name</mat-label>

<input type="text" class="form-control" formControlName="firstName" matInput placeholder="First Name" required>

<mat-error \*ngIf="(UserFirstname.touched) && (UserFirstname.invalid)"></mat-error>

</mat-form-field><br>

</div>

<div class="form-group">

<mat-form-field>

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

<input type="text" class="form-control" formControlName="lastName" required matInput placeholder="Last Name">

<mat-error \*ngIf="(UserLastname.touched) && (UserLastname.invalid)"></mat-error>

</mat-form-field><br>

</div>

<div class="form-group">

<mat-form-field>

<mat-label for="userName">User Name</mat-label>

<input type="text" matInput placeholder="User Name" class="form-control" formControlName="userName" required>

<mat-error \*ngIf="(UserName.touched) && (UserName.invalid)"></mat-error>

</mat-form-field><br>

</div>

<div class="form-group">

<mat-form-field>

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

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

<mat-error \*ngIf="(UserPassword.touched) && (UserPassword.invalid)"></mat-error>

</mat-form-field><br>

</div>

<div class="form-group">

<mat-form-field>

<mat-label for="confirmPassword">Confirm Password</mat-label>

<input type="password" matInput placeholder="Confirm Password" class="form-control" formControlName="confirmPassword" required>

<mat-error \*ngIf="(UserConfirmPassword.touched) && (UserConfirmPassword.invalid)"></mat-error>

</mat-form-field><br>

</div>

<div class="form-group">

<mat-form-field>

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

<input type="email" matInput placeholder="Email" class="form-control" formControlName="email" required>

<mat-error \*ngIf="(UserEmail.touched) && (UserEmail.invalid)"></mat-error>

</mat-form-field><br>

</div>

<div class="form-group">

<mat-form-field>

<mat-label for="role">Choose Your Role</mat-label>

<mat-select>

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

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

<input class="form-control" formControlName="role">

</mat-select>

</mat-form-field><br>

</div>

</mat-card-content>

<mat-card-actions>

<div class="form-group">

<div class="example-label">

<div class="example-button-container">

<button type="submit" mat-raised-button color="primary" class="btn btn-success" formControlName="createUser">Register Now</button>

</div>

</div>

</div>

</mat-card-actions>

</form>

</mat-card>

<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>User Added SuccessFully!</h4>

<button (click)="addRegistrationForm()" class='btn btn-primary'>Add More User</button>

</div>

</div>

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

</div>

Login-user:

<mat-toolbar color="primary" class="container-fluid">

<mat-toolbar-row>

<span>Online Collaborate</span>

<span class="example-spacer"></span>

<section>

<div class="example-button-row">

<div class="example-flex-container">

<button mat-icon-button aria-label="Home page" routerLink="/home">

<mat-icon>home</mat-icon>

</button>

<button mat-icon-button aria-label="Chat" routerLink="/about">

<mat-icon>chat\_bubble</mat-icon>

</button>

<button mat-icon-button aria-label="Search">

<mat-icon>search</mat-icon>

</button>

<button mat-icon-button [matMenuTriggerFor]="menu" class="example-icon" aria-label="Menu">

<mat-icon>menu</mat-icon>

</button>

<mat-menu #menu="matMenu">

<button mat-menu-item routerLink="/login-user">

<mat-icon>login</mat-icon>

<span>Login</span>

</button>

<button mat-menu-item routerLink="/add-user">

<mat-icon>app\_registration</mat-icon>

<span>Sign up</span>

</button>

</mat-menu>

</div>

</div>

</section>

<mat-divider></mat-divider>

</mat-toolbar-row>

</mat-toolbar>

<div class="bgImage">

<mat-card class="example-card">

<mat-card-header>

<div mat-card-avatar class="example-header-image"></div>

<mat-card-title class="card-header">Login</mat-card-title>

<mat-card-subtitle class="card-subtitle">Sign into your Space!!!</mat-card-subtitle>

</mat-card-header>

<mat-card-content>

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

<div>

<mat-form-field class="emailInput">

<mat-label>User Name</mat-label>

<input type="text" class="form-control" name="userName" matInput placeholder="User Name" formControlName="userName">

</mat-form-field>

</div>

<div>

<mat-form-field class="emailInput">

<mat-label>Enter your password</mat-label>

<input type="password" class="form-control" name="password" matInput [type]="hide ? 'password' : 'text'" formControlName="password" >

<button mat-icon-button matSuffix (click)="hide = !hide" [attr.aria-label]="'Hide password'" [attr.aria-pressed]="hide">

<mat-icon>{{hide ? 'visibility\_off' : 'visibility'}}</mat-icon>

</button>

</mat-form-field>

</div>

</form>

</mat-card-content>

<mat-card-actions>

<button type="submit" mat-raised-button color="primary" (click)="validateUser()">Login</button>

</mat-card-actions>

</mat-card>

</div>

* User-Dash:
* <div class="grid-container">
* <h1 class="mat-h1">Dashboard</h1>
* <mat-grid-list cols="2" rowHeight="350px">
* <mat-grid-tile \*ngFor="let card of cards | async" [colspan]="card.cols" [rowspan]="card.rows">
* <mat-card class="dashboard-card">
* <mat-card-content class="dashboard-card-content">
* <div>
* <img src="../../assets/171-1717870\_stockvader-predicted-cron-for-may-user-profile-icon-png.jpg" alt="Avatar">
* </div>
* <div>
* <h2 class="profilename">User's Profile</h2>
* </div>
* <div>
* <h2 class="profilename2">Online Collaborate</h2>
* </div>
* </mat-card-content>
* </mat-card>
* </mat-grid-tile>
* </mat-grid-list>
* </div>
* Home Page:
* <mat-toolbar color="primary" class="container-fluid">
* <mat-toolbar-row>
* <span>Online Collaborate</span>
* <span class="example-spacer"></span>
* <section>
* <div class="example-button-row">
* <div class="example-flex-container">
* <button mat-icon-button aria-label="Home page" routerLink="/home">
* <mat-icon>home</mat-icon>
* </button>
* <button mat-icon-button aria-label="Chat" routerLink="/about">
* <mat-icon>chat\_bubble</mat-icon>
* </button>
* <button mat-icon-button aria-label="Search">
* <mat-icon>search</mat-icon>
* </button>
* <button mat-icon-button [matMenuTriggerFor]="menu" class="example-icon" aria-label="Menu">
* <mat-icon>menu</mat-icon>
* </button>
* <mat-menu #menu="matMenu">
* <button mat-menu-item routerLink="/login-user">
* <mat-icon>login</mat-icon>
* <span>Login</span>
* </button>
* <button mat-menu-item routerLink="/add-user">
* <mat-icon>app\_registration</mat-icon>
* <span>Sign up</span>
* </button>
* </mat-menu>
* </div>
* </div>
* </section>
* <mat-divider></mat-divider>
* </mat-toolbar-row>
* </mat-toolbar>
* <section class="home">
* <img src="../../assets/Homeb.jpg">
* <div class="home1content">
* <h1>Welcome to Online Collaborate</h1>
* <br>
* <br>
* <button style="margin-left: 30%;" mat-raised-button color="primary" class="button" routerLink="/about">About Us</button>
* </div>
* </section>
* <section>
* </section>

**Future Improvements:**

1. Blog will be improved.
2. Functionalities of search bar will be achieved.
3. UI will be improved.
4. Planning to add a Chat on this project.
5. Security Features like end-to-end will be implemented in the future.
6. More Features and functionalities are planned.
7. The name of this Application will be Changed in a period of 6-7 months.
8. You will see a Live Project open to users very soon. So it won’t just be a git-hub project.

**Open to Suggestion’s and You can mail me anytime through the contact form of my Blog,**

**niranninja.com**

**References:**

**I have Used the Angular Material in this project and I took the following references from their official website.**

* [**https://material.angular.io**](https://material.angular.io)
* [**https://material.angular.io/components/form-field/overview**](https://material.angular.io/components/form-field/overview)
* [**https://material.angular.io/components/button/overview**](https://material.angular.io/components/button/overview)
* **https://material.angular.io/components/list/overview**

I took some reference for the dashboard and made something similar to this dashboard

* [**http://angular-material.fusetheme.com/apps/dashboards/analytics**](http://angular-material.fusetheme.com/apps/dashboards/analytics)

This below video have been taken as reference for login,

* [**https://www.youtube.com/watch?v=lolNy59JJT8**](https://www.youtube.com/watch?v=lolNy59JJT8)