***ONLINE COLLABORATION***



**Developed by : Sanjeevi s**

**Reg No :**  R210030100022

**Batch Code :** 210167

**Name of the Cordinator :** Ms. Lopamudra Bera

**Date of Submission :** 14/03/2021

**NIIT**

###### **CERTIFICATE**

This is to certify that this report, titled **ONLINE COLLABORATION** embodies the original work done by **Mr.S.SANJEEVI**, in partial fullfillment of his course requirement at NIIT.

**ACKNOWLEDGEMENT**

Any accomplishment requires work of many people.I am gratefull to our teacher Ms.LOPAMUDRA BERA who guided me to complete this project and my special thanks to my batch members who helped me in completing this project.Learning new things kept me interested in doing this project.Finally i would like to thank my family who supported me throughout the project.

**ABSTRACT**

This project works as an online site for bloggers to post their blog which can be seen by many people around the globe. Anyone can register in this site which will be approved by the Admin and the users are allowed to post their blog in the site which will also be approved by the Admin. Admin has the whole control for the site including the deletion of the users.

### CONFIGURATION

**Hardware:**

* Laptop -Pentium
* 4 Ram,500GB Hard Disk

**Operating System:**

* Windows 10 x64

**Software:**

* Mysql , Eclipse ,Visual Studio

**AIM:**

To create an online application for the bloggers to blog around the globe.

**OBJECTIVES:**

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

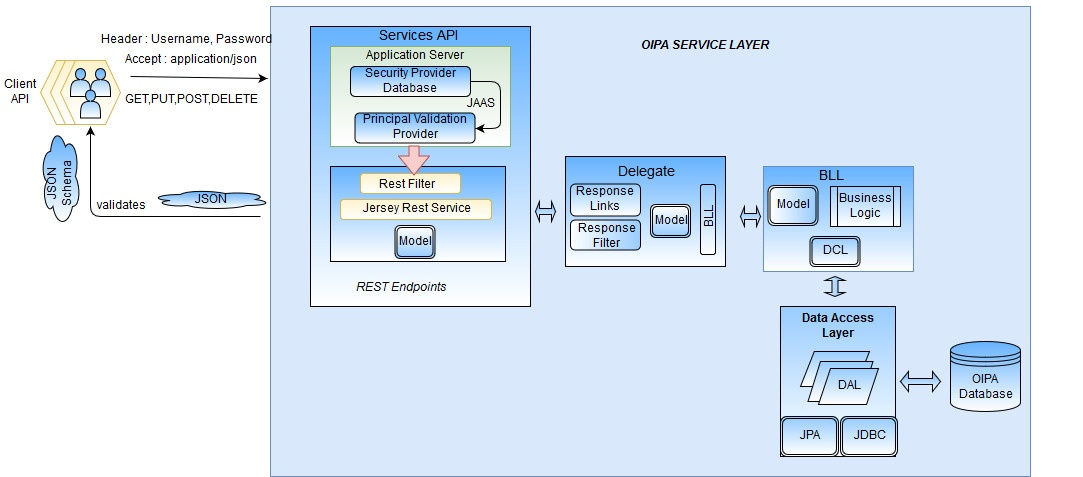
**Software Requirements :**

* ECLIPSE
* MYSQL
* VISUAL STUDIO
* MS WORD
* MS EXCEL

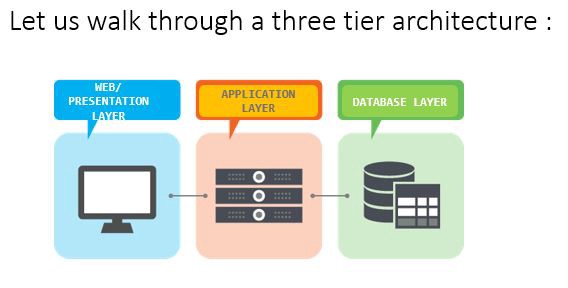
**PROJECT PLAN**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **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**



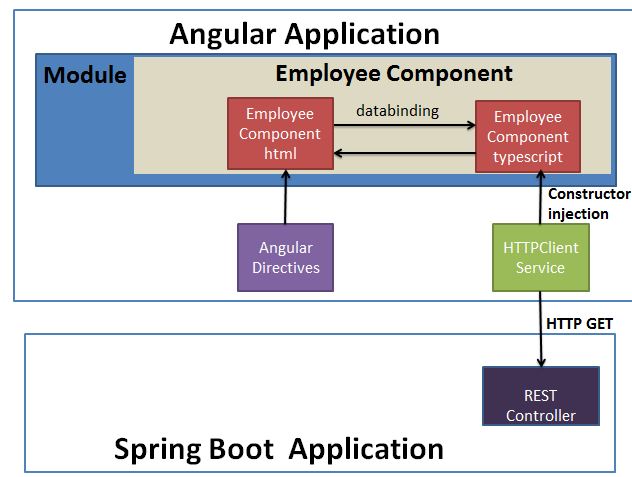
### SYSTEM ARCHITECTURE

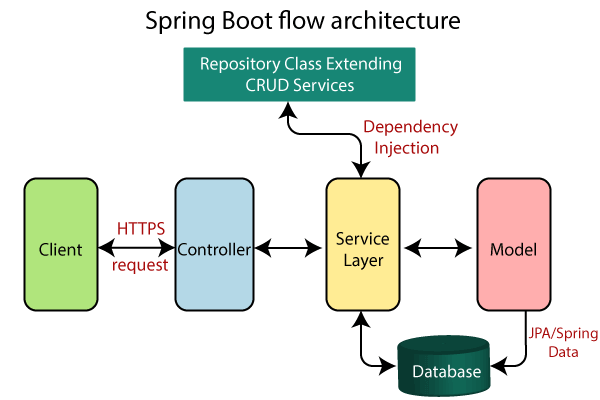


1. Tier Architecture

* **Presentation Tier:**
* The presentation tier is the front end layer in the 3-tier system and consists of the user interface. This user interface is often a graphical one accessible through a web browser or web-based application and which displays content and information useful to an end user. This tier is often built on web technologies such as HTML5, JavaScript, CSS, or through other popular web development frameworks, and communicates with others layers through API calls.
* **Application Tier:**
* The application tier contains the functional business logic which drives an application’s core capabilities. It’s often written in Java, .NET, C#, Python, C++, etc.
* **DataBase Tier:**
* The data tier comprises of the database/data storage system and data access layer. Examples of such systems are MySQL, Oracle, PostgreSQL, Microsoft SQL Server, MongoDB, etc. Data is accessed by the application layer via API calls.

**BUSINESS PROCESS MODEL**





**PROJECT SETUP**

MySQL download-install-setup:

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

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

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

Project:

1. Create a SpringBoot project named onlineCollaborate with web, Spring Data JPA, SpringBoot Dev Tools and MySQL Server Driver packages. Extract that project.
2. Import the project in Eclipse.
3. Create the configuration class  
   Instead of XML, we perform annotation-based configuration. So, we create a class Config.java and specify the required configuration in it. However, there is one more configuration class OnlineCollaborate.java. This class is provided by Spring Boot automatically.
4. Create the entity class  
   Here, we are creating an Entity/POJO (Plain Old Java Object) class.

**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);

}

}

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

ng new OnlineCollaborateAngular

11.Install Bootstrap CSS framework

Use the following command to install bootstrap in the project.

*D:\All\_Workspace\Angular\_Workspace\SrudentAngular>****npm install*** [***bootstrap@3.3.7***](mailto:bootstrap@3.3.7) ***–save***

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

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

13.Install Angular Data Table

Use the following command to install bootstrap in the project.

*PS D:\All\_Workspace\Angular\_Workspace\OnlineCollaborateAngular>****npm install angular-datatable –save***

14.It is required to include **DataTableModule** in imports array of **app.module.ts** file

15. Generate Components  
Open the project in visual studio and then use the following command to generate Angular components:

ng g c RegisterUser

ng g c UserList

ng g c LoginUser

ng g c ActiveUser

ng g c BlogList

ng g c Home

ng g c UserProfile

ng g c Navigate

ng g c NavigateAdmin

ng g c NavigateUser

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

ng g s user

ng g s blog

17.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

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

User.ts

export class User {

   userId!: number;

   firstName!: String;

   lastName!:String;

   username!:String;

   password!:String;

   confirm\_password!:String;

   email!:String;

   role!:String;

   status!:String;

   isOnline!:boolean;

   enabled!:boolean;

}

Blog.ts

export class Blog{

    blogId!:number;

    blogTitle!:String;

    blogContent!:String;

    blogPosted!:any;

    status!:String;

    noOfLikes!:number;

    noFoComments!:number;

    noOfViews!:number;

    userId!:number;

    username!:String;

}

22. Edit the **service.ts** file

User.service.ts

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

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

import { Observable } from 'rxjs';

@Injectable({

  providedIn: 'root'

})

export class UserService {

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

  constructor(private http:HttpClient) {

   }

  getUserList(): Observable<any> {

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

      }

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

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

          }

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

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

          }

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

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

          }

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

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

          }

          deactiveList():Observable<any> {

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

          }

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

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

              }

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

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

              }

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

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

                  }

}

Blog.service.ts

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

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

import { Observable } from 'rxjs';

@Injectable({

  providedIn: 'root'

})

export class BlogService {

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

  constructor(private http:HttpClient){ }

  getBlogList(): Observable<any>{

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

  }

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

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

  }

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

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

  }

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

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

  }

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

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

  }

}

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

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

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

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

import { User } from '../user';

@Component({

  selector: 'app-register-user',

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

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

})

export class RegisterUserComponent implements OnInit {

  constructor(private userservice:UserService) { }

  user : User=new User();

  submitted = true;

  ngOnInit(): void {

    this.submitted=false;

  }

  registrationform=new FormGroup({

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

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

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

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

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

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

    role:new FormControl()

      });

      register(){

        if(this.registrationform.valid){

        this.user=new User();

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

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

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

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

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

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

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

        if(this.user.role==="Admin"){

          this.user.enabled=true;

          this.user.status="Active";

        }

        else{

          this.user.enabled=false;

        this.user.status="Inactive";

        }

        this.user.isOnline=false;

        this.submitted=true;

        console.log(this.user.firstName);

        this.save();

      }

      }

      save(){

        this.userservice.createUser(this.user)

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

          this.user=new User();

      }

      get FirstName(){

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

      }

      get lastName(){

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

      }

      get Username(){

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

      }

      get Password(){

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

      }

      get ConfirmPassword(){

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

      }

      get Email(){

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

      }

      get Role(){

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

      }

      registrationForm(){

        this.submitted=false;

        this.registrationform.reset();

      }

      logInUser() {

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

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

      }

}

Register-user.html

        <h2 style="font-weight: bold;text-align:center;"><I>REGISTRATION FORM</I></h2>

<div class="row">

   <div class="col-sm-4"></div>

   <div class="col-sm-4" >

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

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

                        <div>

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

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

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

                    </div>

               <div>

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

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

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

                </div>

                <div >

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

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

                     title="Enter Username" placeholder="Enter Your Username">

                </div>

                 <div >

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

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

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

                 </div>

                 <div >

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

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

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

                 </div>

                 <div >

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

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

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

                 </div>

                  <div >

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

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

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

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

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

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

                      </select>

                  </div>

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

            </form>

         </div>

   </div>

    <div [hidden]="!submitted">

        <h4 style="font-weight: bold; font-style:italics; text-align:center;color:pink;">Congratulations! You have registered succesfully!</h4>

 </div>

User-list.component.ts

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

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

import { User } from '../user';

import { Observable,Subject } from "rxjs";

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

import {DataTablesModule} from 'angular-datatables';

@Component({

selector: 'app-user-list',

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

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

})

export class UserListComponent implements OnInit {

constructor(private userservice:UserService) { }

usersArray: any[] =[];

dtOptions: DataTables.Settings = {};

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

users!: any;

user : User=new User();

deleteMessage=false;

userlist:any;

isupdated = false;

ngOnInit() {

this.isupdated=false;

this.dtOptions = {

pageLength: 6,

stateSave:true,

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

processing: true

    };

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

this.users =data;

this.dtTrigger.next();

    })

  }

deleteUser(userId: number) {

this.userservice.deleteUser(userId)

      .subscribe(

(data) => {

console.log(data);

this.deleteMessage=true;

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

this.users =data

            })

        },

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

  }

updateUser(userId: number){

this.userservice.getUser(userId)

      .subscribe(

(data) => {

this.userlist=data;

console.log(this.userlist);

        }),

(error:any) => console.log(error);

  }

userupdateform=new FormGroup({

userId:new FormControl(),

firstName:new FormControl(),

lastName:new FormControl(),

username:new FormControl(),

password:new FormControl(),

email:new FormControl(),

role:new FormControl(),

status:new FormControl(),

isOnline:new FormControl(),

enabled:new FormControl()

  });

updateUsers(updateUser:any){

  this.user=new User();

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

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

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

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

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

  this.user.email!=this.Email!.value;

  this.user.role!=this.Role!.value;

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

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

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

  console.log(this.FirstName!.value);

  console.log(this.Role!.value);

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

    data => {

    this.isupdated=true;

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

    this.users =data

            })

        },

        error =>{

          console.log(this.users);

           console.log(error)});

      }

get FirstName(){

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

}

get LastName(){

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

}

get UserName(){

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

}

get Password(){

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

}

get Email(){

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

}

get Role(){

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

}

get Status(){

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

}

get IsOnline(){

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

}

get Enabled(){

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

}

get UserId(){

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

}

changeisUpdate(){

  this.isupdated=false;

}

}

User-list.component.html

<div class="panel panel-default">

  <div class="panel-heading">

      <h1 style="font-weight: bold;text-align: center;"><I>ALL USERS</I></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">x</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>

          <th>Action</th>

         </tr>

        </thead>

        <tbody>

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

          <td>{{user.userId}}</td>

          <td>{{user.firstName}}</td>

          <td>{{user.lastName}}</td>

          <td>{{user.username}}</td>

          <td>{{user.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)="updateUsers(user.userId)" 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" #updateusers (ngSubmit)="updateUsers(updateUser)">

            <!-- Modal Header -->

            <div class="modal-header">

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

            </div>

             <!-- Modal body -->

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

               <div [hidden]="isupdated">

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

                  <div >

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

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

                  </div>

                  <div >

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

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

                  </div>

                  <div >

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

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

                  </div>

                  <div >

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

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

                  </div>

                  <div >

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

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

                  </div>

                  <div >

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

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

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

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

                      </select>

                  </div>

                  <div >

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

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

                  </div>

                  <div >

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

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

                  </div>

                  <div >

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

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

                  </div>

                </div>

                <div [hidden]="!isupdated">

                   <h4>User Detail Updated!</h4>

                </div>

            </div>

            <!-- Modal footer -->

            <div class="modal-footer" >

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

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

            </div>

           </form>

        </div>

    </div>

</div>

active-user.component.ts

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

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

import  {User } from '../user';

import { Observable,Subject } from 'rxjs';

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

import { DataTablesModule } from 'angular-datatables';

@Component({

  selector: 'app-active-user',

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

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

})

export class ActiveUserComponent implements OnInit {

  dtOptions:DataTables.Settings={};

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

  constructor(private userservice:UserService) { }

  users!:any;

  user:User=new User();

  deactiveList:any;

  isEnabled=false;

  ngOnInit(): void {

    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();

    })

  }

  enableUser(id:number){

    this.userservice.activeUser(id).subscribe(

      ( data)=>{

        this.deactiveList=data;

        console.log(this.deactiveList);

        this.activeUser(id);

      },

      ( error) =>console.log(error)

    );

  }

  activeUserForm= new FormGroup({

    userId:new FormControl()

  });

  activeUser(id:number){

    this.user=new User();

    this.user.userId=id;

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

      (      data )=>{

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

          this.users=data

          console.log(this.users)

        })

      },

      (error)=>console.log(error)

    );

  }

  get UserId(){

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

  }

}

active-user.component.html

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

                        <th>Enabled</th>

                        <th>Role</th>

                        <th>Status</th>

                        <th>Action</th>

                    </tr>

                </thead>

                <tbody>

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

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

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

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

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

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

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

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

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

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

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

                    </tr>

                </tbody>

                </table>

            </div>

    </div>

</div>

login-user.component.ts

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

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

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

import { data } from 'jquery';

import { User } from '../user';

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

import { Observable , Subject } from 'rxjs';

@Component({

  selector: 'app-login-user',

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

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

})

export class LoginUserComponent implements OnInit {

  user :User=new User();

  currentUser :any;

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

  ngOnInit(): void {

  }

  loginform=new FormGroup({

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

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

  })

  validateUser(){

    this.user=new User();

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

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

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

      (      data) =>{

        console.log(data);

        if(data!=null){

          this.currentUser=data;

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

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

        }

        else{

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

        }

      }

      else {

        console.log("Object Empty");

      }

    },

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

    )

  }

  get Username(){

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

  }

  get Password(){

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

  }

  RegUser(){

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

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

  }

}

login-user.component.html

    <div class="login-box">

        <h2>Login</h2>

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

            <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>

    <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>

User-profile.component.ts

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

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

import { User } from '../user';

import { Observable, Subject } from 'rxjs';

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

import { DataTablesModule } from 'angular-datatables';

import { param } from 'jquery';

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

@Component({

  selector: 'app-user-profile',

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

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

})

export class UserProfileComponent implements OnInit {

  Id!:number;

  user:any;

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

  ngOnInit(): 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);

          }

     )

      }

     )

  }

}

User-profile.component.html

<div>

    <div class="main">

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

          <div class="padding">

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

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

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

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

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

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

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

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

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

                                  </div>

                              </div>

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

                                  <div class="card-block">

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

                                      <div class="row">

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

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

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

                                          </div>

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

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

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

                                          </div>

                                      </div>

                                  </div>

                              </div>

                          </div>

                      </div>

                  </div>

              </div>

          </div>

      </div>

      </div>

  </div>

Home.component.ts

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

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

@Component({

  selector: 'app-home',

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

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

})

export class HomeComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {

  }

  logInUser() {

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

  }

  RegUser(){

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

  }

}

[Home.component.html](http://Home.component.html)

<div class="container">

    <div class="navbar">

        <div class="menu">

        </div>

    </div>

    <div class="main-container">

        <div class="main">

            <header>

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

                    <h2 class="title"><span style="color:greenyellow">CONNECT TO BLOG</span> </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">REGISTER</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>

Navigate.component.ts

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

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

import { Observable } from 'rxjs';

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

@Component({

  selector: 'app-nav',

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

  styleUrls: ['./nav.component.scss']

})

export class NavComponent implements OnInit {

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

  .pipe(

    map(result => result.matches),

    shareReplay()

  );

  constructor(private breakpointObserver:BreakpointObserver) { }

  ngOnInit(): void {

  }

}

Navigate.component.html

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

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

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

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

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

      <mat-toolbar>Menu</mat-toolbar>

      <mat-nav-list>

        <a mat-list-item routerLink="user-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;">CONNECT TO BLOG</span>

      </mat-toolbar>

    </mat-sidenav-content>

  </mat-sidenav-container>

Navigate-admin.component.ts

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

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

import { Observable } from 'rxjs';

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

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

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

import { param } from 'jquery';

@Component({

  selector: 'app-navigation',

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

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

})

export class NavigationComponent implements OnInit {

  Id!: number;

  user:any;

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

   .pipe(

     map(result => result.matches),

     shareReplay()

   );

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

  ngOnInit(): 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)

    );

 }

}

Navigate-admin.component.html

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

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

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

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

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

      <mat-toolbar>MENU</mat-toolbar>

      <mat-nav-list>

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

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

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

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

        </mat-nav-list>

    </mat-sidenav>

    <mat-sidenav-content>

      <mat-toolbar color="primary">

        <span style="font-style: italic;">CONNECT TO BLOG</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>

Navigate-user.component.ts

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

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

import { Observable } from 'rxjs';

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

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

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

import { param } from 'jquery';

@Component({

  selector: 'app-navigation-user',

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

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

})

export class NavigationUserComponent implements OnInit {

  Id!:number;

  user:any;

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

    .pipe(

      map(result => result.matches),

      shareReplay()

    );

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

  ngOnInit(): 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)

    );

 }

}

Navigate-user.component.html

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

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

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

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

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

      <mat-toolbar>MENU</mat-toolbar>

      <mat-nav-list>

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

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

      </mat-nav-list>

    </mat-sidenav>

    <mat-sidenav-content>

      <mat-toolbar color="primary">

        <span>CONNECT TO BLOG</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>

28. Install following:

npm install jquery --save

npm install datatables.net --save

npm install datatables.net-dt --save

npm install angular-datatables@6.0.0 --save

npm install @types/jquery --save-dev

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

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

"styles": [

"src/styles.scss",

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

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

],

"scripts": [

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

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

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

]

30.Mysql DATABASE:

**USER TABLE:**

create table User(

UserId int not null auto\_increment,

FirstName varchar(30),

LastName varchar (30),

UserName varchar (30),

Password varchar (30),

Email varchar(40),

Role varchar(5),

Status varchar(10),

IsOnline boolean,

Enabled boolean,

primary key (UserId)

);

**BLOG TABLE:**

create table Blog(

BlogId int not null auto\_increment,

BlogTitle varchar(20),

BlogContent varchar(300),

BlogPosted date,

Status varchar(10),

NoOfLikes int,

NoOfComments int,

NoOfViews int,

UserId int,

UserName varchar(20),

primary key(BlogId)

); **BLOGCOMMENTS TABLE:**

create table BlogComments(

BlogCommentsId int not null auto\_increment,

UserId int,

UserName varchar(20),

ProfileId int,

Title varchar(20),

NoOfLikes int,

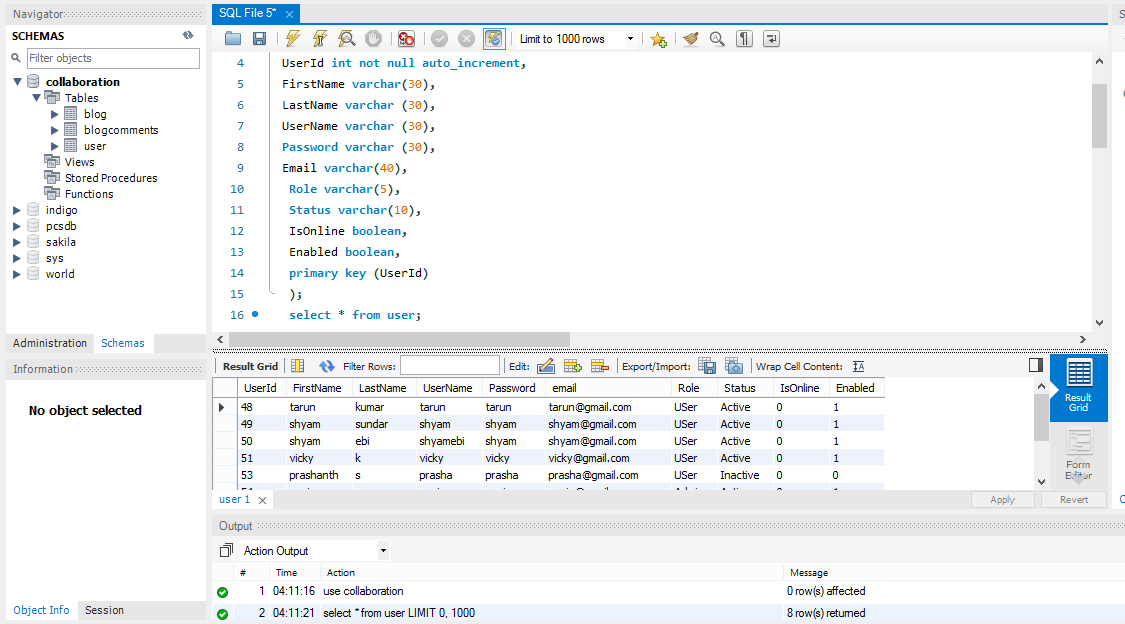
BlogComments varchar(300),

CurrentDate date,

BlogId int,

primary key(BlogCommentsId)

);



3.Save All

.

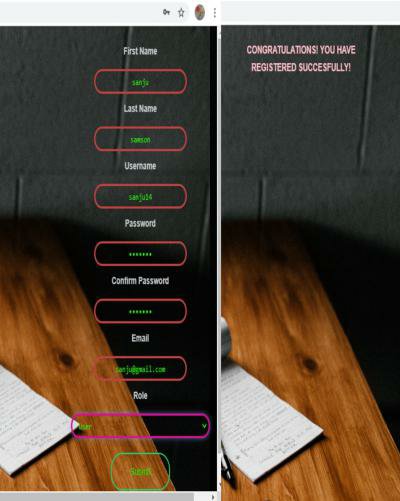
31.Run the SpringBoot Application

32.Run the Angular project

**LANDING PAGE:**

****

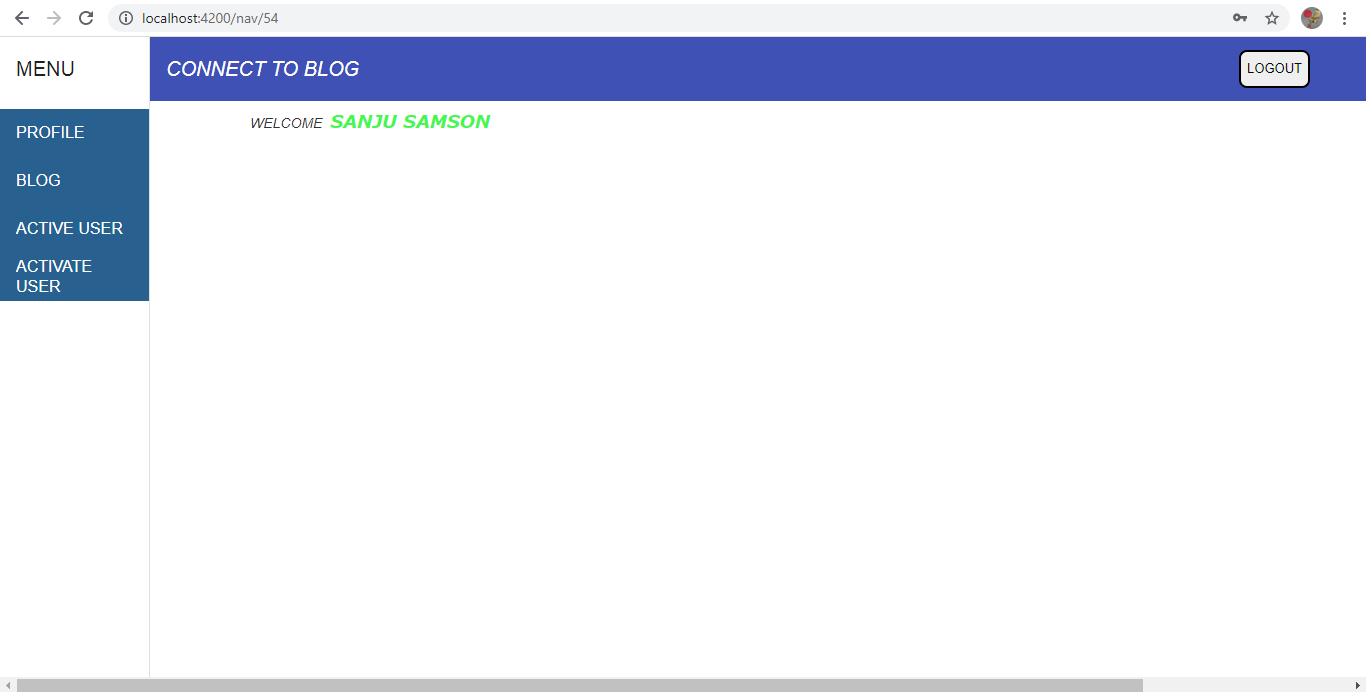
**REGISTRATION PAGE:**

****

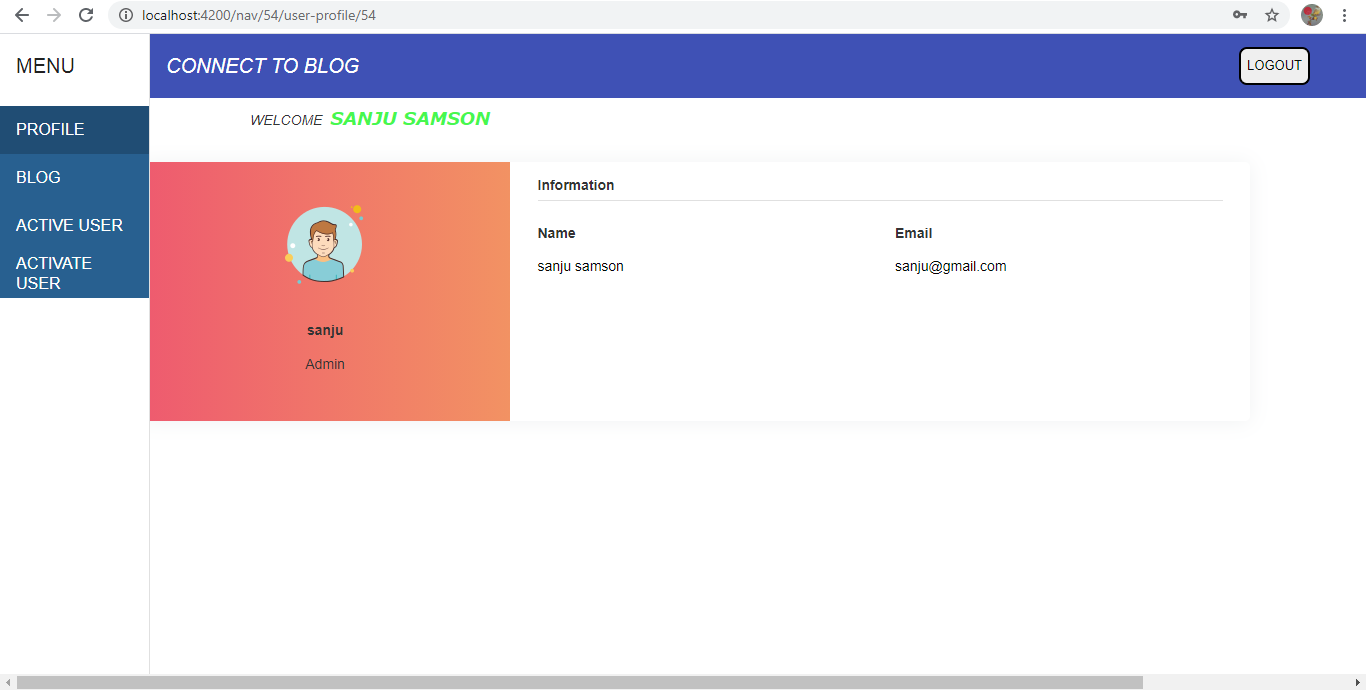
**LOGIN PAGE:**

****

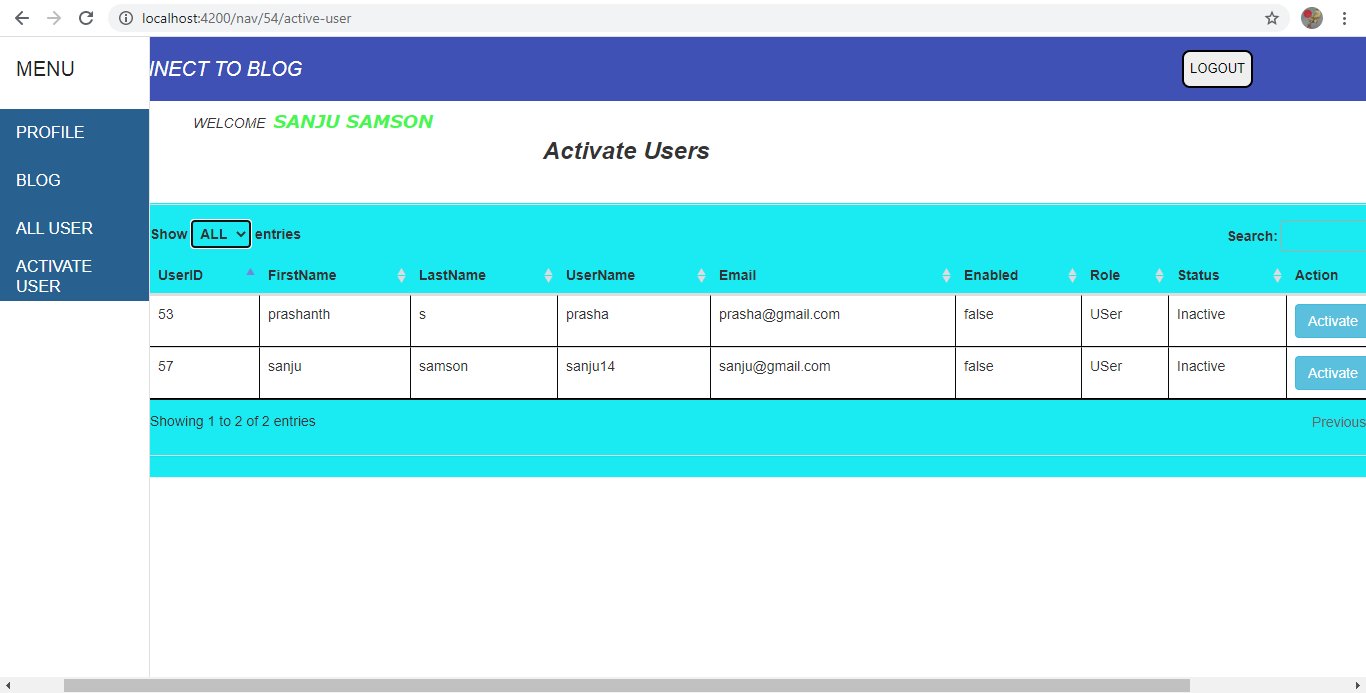
**WELCOME PAGE:**



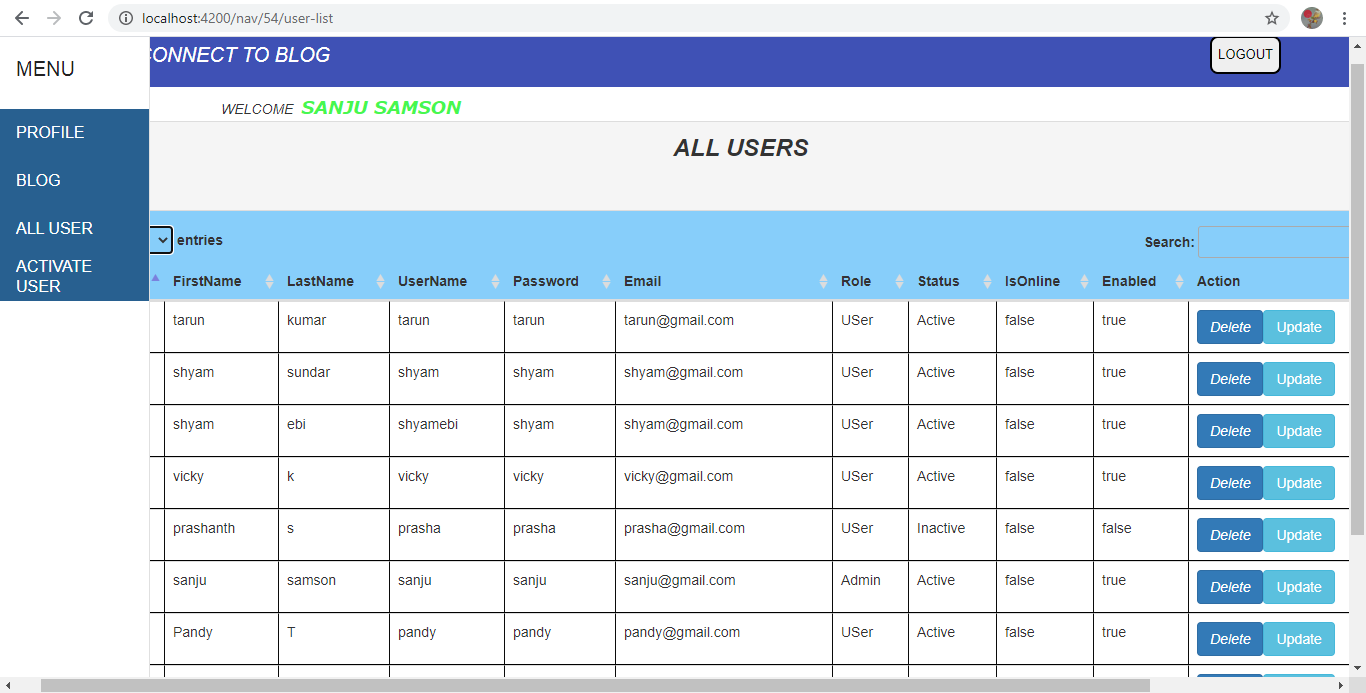
**PROFILE PAGE:**

****

**ACTIVATE USER PAGE:**

****

**ALL USER PAGE:**

****