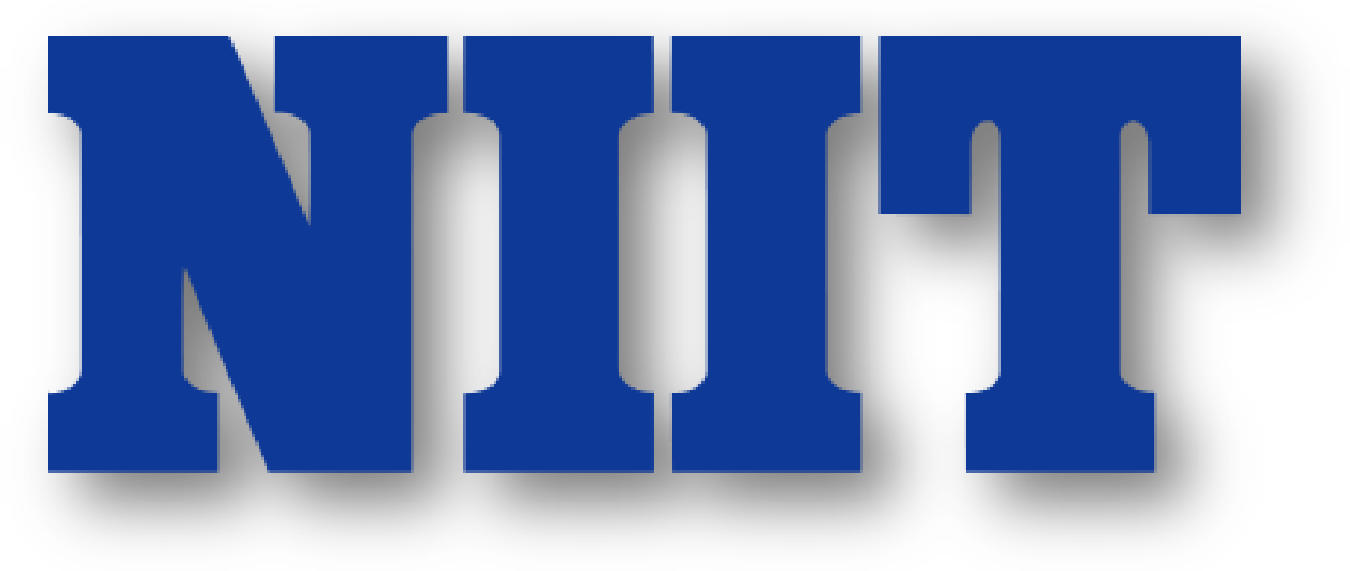
PROJECT ON

ONLINE COLLABORATION

DEVELOPED By –

KAMISHETTY ABHILASH



ONLINE COLLABORATION

Batch Code: S210192

Start Date : 2020

End Date :

Name of the

Coordinator : Mrs. Lopamudra Bera

Name of the

Developer : KAMISHETTY ABHILASH

Date of

Submission : 14/04/2021

**ONLINE COLLABORATION**

CERTIFICATE

This is to certify that this report, ONLINECOLLABORATION Application embodies the original work done by KAMISHETTY ABHILASH in partial fulfillment of their course requirement at NIIT.

CO-ORDINATOR:

### Abstract

This project manages the entire process of Blogging website to help Bloggers and Users to use the site with ease.

Its also provides single window system to Blogger

who can showcase their ideas and skills in their respective fields or showcase their passion.

### Configuration

**Hardware**

HP(i3 processor),

8Gb RAM,

1TB Hard Disk.

**Operating System**

Windows 10

**Software**

* Eclipse IDE
* Java
* Spring Boot
* Hibernate
* MySQL
* Visual Studio
* Angular
* Html
* SCSS
* Typescript

### TABLE OF CONTENTS

Chapter 1- Introduction

1.0 Aim

1.1 Objectives

Chapter 2- Project Requirement Specification

2.0 Literature Research

2.1 Statement of Requirements

Chapter 3- Project Analysis

3.0 Project Plan

3.1 System Architecture

3.2 Business Process Model

3.3 Software Requirement Specification

3.4 High Level Use Case Diagrams

Chapter 4- Project Design

4.0 Low Level Use Case Diagrams

4.1 User Interface Design

4.2 Systems Input and Output Design

4.3 Database Structure

4.4 Sample Code

Challenges

Observations

References

Appendix

## AIM

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

### OBJECTIVES

Following are the objectives to be achieved through Online Collaboration Application

* User registration
* Profile Validation
* Activation/Approving
* Posting Blogs
* View Blogs

This objectives are to be automated which were previously done manually.

### STATEMENT OF REQUIREMENTS

PROJECT OBJECTIVES

* **About Your Company**
  + Online Collaborate is a public collaboration website which provides a platform to connect with people to share their works.
* **Need for Collaboration**

### It helps us problem-solve

### Collaboration brings people and organizations closer

### Collaboration helps people learn from each other

### It opens up new channels for communication

### Collaboration boosts morale across your organization

### It leads to higher retention rates

### Collaboration makes us more efficient workers

### PROJECT LIFE CYCLE MODEL



### SYSTEM ARCHITECTURE

1. Tier Architecture
2. A **Presentation Layer** that sends content to browsers in the form of HTML/JS/CSS. This might leverage frameworks like React, Angular, Ember, Aurora, etc.
3. An **Application Layer** that uses an application server and processes the business logic for the application. This might be written in C#, Java, C++, Python, Ruby, etc.
4. A **Data Layer** which is a database management system that provides access to application data. This could be MSSQL, MySQL, Oracle, or PostgreSQL, Mongo, etc.

**Online Collaboration**

**Presentation Layer**

Home

**Angular Component**

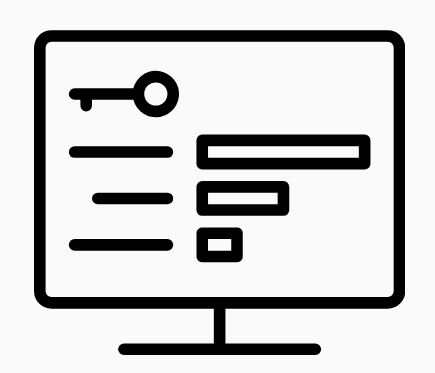
(Login,Registration,User-list

Blog-List,Service)

ADMIN

USER





**JRE**

**DATABASE**

(Tables-User,Blog,BlogComments,)

Service

Controller

Configuration

CRUD OP

**Hibernate Config**

SOFTWARE REQUIREMNETS SPECIFICATION

Table of Contents

1. Introduction
2. Definitions,acronyms and abbreviations
   1. References
   2. Overview
3. General Description

2.1 Product perspective

2.2 Product functions

2.3 User characteristics

2.4 Constraints

2.5 Assumption and dependencies

1. Specific Requirements

3.1 Functional requirements

3.2 Non-functional requirements

3.3 External interface requirements

3.4 Performance requirements

3.5 Design constraints

3.6 Attributes

3.7 Other requirements

1. Appendices
2. Index

#### HIGH LEVEL USER CASE DIAGRAMS

Validation

Home

NAV

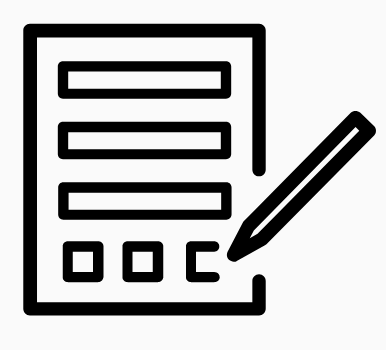
Admin

User

Sign Up

Sign In

**Registration Form**



User

Admin

### 

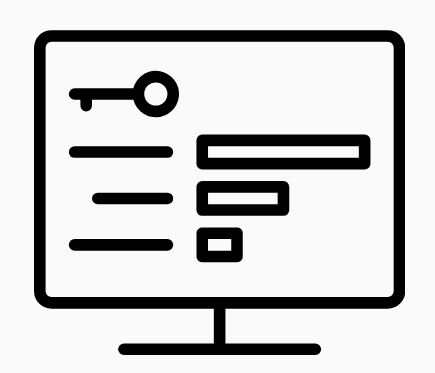




Blogging

Viewing





Activation

Approving

DATABASE

Hibernate config

CRUD/BackEnd

FrontEnd

#### LOW LEVEL USER CASE DIAGRAMS

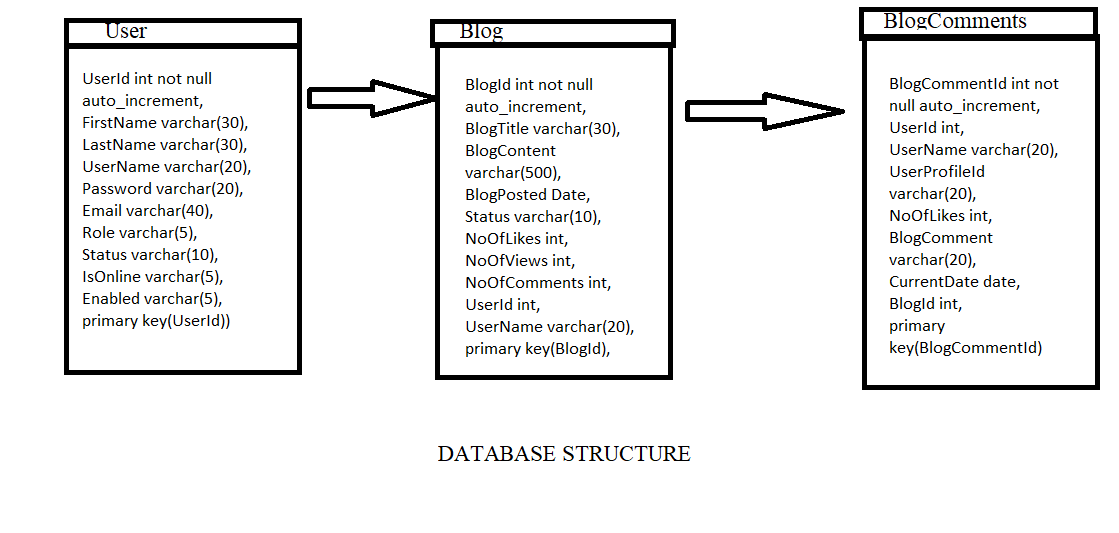
Database

Layer

Application

Layer

User Interface Layer



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>

Database:

use collaborate scheme

1.Created a table User.

create table User(

UserId int not null auto\_increment,

FirstName varchar(30),

LastName varchar(30),

Username varchar(20),

Password varchar(20),

Email varchar(40),

Role varchar(5),

Status varchar(10),

IsOnline boolean,

Enabled boolean,

primary key(UserId)

)

2.Created a table Blog.

create table Blog(

BlogId int not null auto\_increment,

BlogTitle varchar(30),

BlogContent varchar(200),

BlogPosted Date,

status varchar(10),

NoOfLikes int,

NoOfViews int,

NoOfComments int,

UserId int,

Username varchar(20),

primary key(BlogId)

)

3.Created a table BlogComments

create table BlogComments(

BlogCommentId int not null auto\_increment,

UserId int,

Username varchar(20),

UserProfileId varchar(20),

Title varchar(30),

NoOfLikes int,

BlogComment varchar(50),

CurrentDate Date,

BlogId int,

primary key(BlogCommentId)

)

Backend Project

1. Create a SpringBoot project named “OnlineCollaborate”(You can suggest any fesible project name according to project specification) 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 HibernateConfig.java inside com.coll.OnlineCollaborate.config package and specify the required configuration in it. However, there is one more configuration class OnlineCollaborateApplication.java. This class is provided by Spring Boot automatically.

package com.coll.OnlineCollaborate.config;

import java.util.Properties;

import javax.sql.DataSource;

import org.springframework.boot.autoconfigure.EnableAutoConfiguration;

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

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.context.annotation.ComponentScans;

import org.springframework.context.annotation.Configuration;

import org.springframework.jdbc.datasource.DriverManagerDataSource;

import org.springframework.orm.hibernate5.HibernateTransactionManager;

import org.springframework.orm.hibernate5.LocalSessionFactoryBean;

import org.springframework.orm.hibernate5.LocalSessionFactoryBuilder;

import org.springframework.transaction.annotation.EnableTransactionManagement;

import org.springframework.web.servlet.ViewResolver;

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

@Configuration

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

@ComponentScan("model"),

@ComponentScan("controller"),

@ComponentScan("dao"),

@ComponentScan("service")})

@EnableAutoConfiguration(exclude = { HibernateJpaAutoConfiguration.class})

@EnableTransactionManagement

public class HibernateConfig {

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

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

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

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

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

@Bean(name="dataSource")

public DataSource getDataSource() {

DriverManagerDataSource dataSource=new DriverManagerDataSource();

dataSource.setDriverClassName(DATABASE\_DRIVER);

dataSource.setUrl(DATABASE\_URL);

dataSource.setUsername(DATABASE\_USERNAME);

dataSource.setPassword(DATABASE\_PASSWORD);

return dataSource;

}

@Bean

public LocalSessionFactoryBean getSessionFactory() {

LocalSessionFactoryBean sessionFactory = new LocalSessionFactoryBean();

sessionFactory.setDataSource(getDataSource());

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

Properties hibernateProperties = new Properties();

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

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

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

sessionFactory.setHibernateProperties(hibernateProperties);

return sessionFactory;

}

@Bean

public HibernateTransactionManager getTransactionManager() {

HibernateTransactionManager txm=new HibernateTransactionManager();

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

return txm;

}

@Bean

public ViewResolver jspViewResolver() {

InternalResourceViewResolver viewResolver=new InternalResourceViewResolver();

viewResolver.setPrefix("/views/");

viewResolver.setSuffix(".jsp");

return viewResolver;

}

}

1. Create Entity(Model) classes. Here, we are creating an Entity/POJO (Plain Old Java Object) class inside com.coll.OnlineCollaborate.model package.
   1. Create a class inside the above said package named “User”.

package com.coll.OnlineCollaborate.model;

import java.io.Serializable;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Transient;

import org.springframework.stereotype.Component;

@Component

@Entity

public class User extends DomainResponse implements Serializable{

private static final long serialVersionUID = 1L;

@Id

@GeneratedValue(strategy=GenerationType.IDENTITY)

private int userId;

private String firstName;

private String lastName;

private String username;

private String password;

private String email;

private String role;

private String status;

private 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***;

}

}

* 1. Create another class inside the above said package named “Domain Response”.

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

**public** **class** DomainResponse {

**int** responseCode;

String responseMessage;

**public** DomainResponse() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** DomainResponse(**int** responseCode, String responseMessage) {

**super**();

**this**.responseCode = responseCode;

**this**.responseMessage = responseMessage;

}

**public** **int** getResponseCode() {

**return** responseCode;

}

**public** **void** setResponseCode(**int** responseCode) {

**this**.responseCode = responseCode;

}

**public** String getResponseMessage() {

**return** responseMessage;

}

**public** **void** setResponseMessage(String responseMessage) {

**this**.responseMessage = responseMessage;

}

}

* 1. Create another class inside the above said package named “Blog”.

package com.coll.OnlineCollaborate.model;

import java.io.Serializable;

import java.time.LocalDate;

import java.util.List;

import javax.persistence.CascadeType;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.OneToMany;

import org.springframework.stereotype.Component;

import com.fasterxml.jackson.annotation.JsonManagedReference;

@Component

@Entity

public class Blog extends DomainResponse implements Serializable{

private static final long serialVersionUID = 1L;

@Id

@GeneratedValue(strategy=GenerationType.IDENTITY)

int blogId;

String blogTitle, blogContent;

LocalDate blogPosted;

String status;

int noOfLikes, noOfComments, noOfViews;

int userId;

String username;

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

@JsonManagedReference

List<BlogComments> blogComments;

**public** **int** getBlogId() {

**return** blogId;

}

**public** **void** setBlogId(**int** blogId) {

**this**.blogId = blogId;

}

**public** String getBlogTitle() {

**return** blogTitle;

}

**public** **void** setBlogTitle(String blogTitle) {

**this**.blogTitle = blogTitle;

}

**public** String getBlogContent() {

**return** blogContent;

}

**public** **void** setBlogContent(String blogContent) {

**this**.blogContent = blogContent;

}

**public** LocalDate getBlogPosted() {

**return** blogPosted;

}

**public** **void** setBlogPosted(LocalDate blogPosted) {

**this**.blogPosted = blogPosted;

}

**public** String getStatus() {

**return** status;

}

**public** **void** setStatus(String status) {

**this**.status = status;

}

**public** **int** getNoOfLikes() {

**return** noOfLikes;

}

**public** **void** setNoOfLikes(**int** noOfLikes) {

**this**.noOfLikes = noOfLikes;

}

**public** **int** getNoOfComments() {

**return** noOfComments;

}

**public** **void** setNoOfComments(**int** noOfComments) {

**this**.noOfComments = noOfComments;

}

**public** **int** getNoOfViews() {

**return** noOfViews;

}

**public** **void** setNoOfViews(**int** noOfViews) {

**this**.noOfViews = noOfViews;

}

**public** **int** getUserId() {

**return** userId;

}

**public** **void** setUserId(**int** userId) {

**this**.userId = userId;

}

**public** String getUsername() {

**return** username;

}

**public** **void** setUsername(String username) {

**this**.username = username;

}

**public** List<BlogComments> getBlogComments() {

**return** blogComments;

}

**public** **void** setBlogComments(List<BlogComments> blogComments) {

**this**.blogComments = blogComments;

}

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

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

}

}

* 1. Create another class inside the above said package named “BlogComments”.

package com.coll.OnlineCollaborate.model;

import java.io.Serializable;

import java.time.LocalDate;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import org.springframework.stereotype.Component;

import com.fasterxml.jackson.annotation.JsonBackReference;

@Component

@Entity

public class BlogComments implements Serializable{

private static final long serialVersionUID = 1L;

@Id

@GeneratedValue(strategy=GenerationType.IDENTITY)

int blogCommentId;

int userId;

String username;

String userProfileId;

String title;

int noOfLikes;

String blogComment;

LocalDate currentDate;

@ManyToOne

@JoinColumn(name="BlogId")

@JsonBackReference

Blog blog;

**public** **int** getBlogCommentId() {

**return** blogCommentId;

}

**public** **void** setBlogCommentId(**int** blogCommentId) {

**this**.blogCommentId = blogCommentId;

}

**public** **int** getUserId() {

**return** userId;

}

**public** **void** setUserId(**int** userId) {

**this**.userId = userId;

}

**public** String getUsername() {

**return** username;

}

**public** **void** setUsername(String username) {

**this**.username = username;

}

**public** String getUserProfileId() {

**return** userProfileId;

}

**public** **void** setUserProfileId(String userProfileId) {

**this**.userProfileId = userProfileId;

}

**public** String getTitle() {

**return** title;

}

**public** **void** setTitle(String title) {

**this**.title = title;

}

**public** **int** getNoOfLikes() {

**return** noOfLikes;

}

**public** **void** setNoOfLikes(**int** noOfLikes) {

**this**.noOfLikes = noOfLikes;

}

**public** String getBlogComment() {

**return** blogComment;

}

**public** **void** setBlogComment(String blogComment) {

**this**.blogComment = blogComment;

}

**public** LocalDate getCurrentDate() {

**return** currentDate;

}

**public** **void** setCurrentDate(LocalDate currentDate) {

**this**.currentDate = currentDate;

}

**public** Blog getBlog() {

**return** blog;

}

**public** **void** setBlog(Blog blog) {

**this**.blog = blog;

}

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

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

}

}

1. Create the DAO interfaces inside com.coll.OnlineCollaborate.dao package:
   1. Create an Interface named IUserDao.java

package com.coll.OnlineCollaborate.dao;

import java.util.List;

import com.coll.OnlineCollaborate.model.User;

public interface IUserDao {

List<User> userListbyStatus(String status);

List<User> getAllUsers();

User getUserById(int userId);

User getUserByUsername(String username);

User validateUser(User user);

boolean addUser(User user);

boolean updateUser(User user);

boolean deleteUser(int userId);

boolean deactiveUser(int userId);

boolean updateUserProfile(String file, Integer userId);

}

* 1. Create an Interface named IBlogDao.java

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);

List<Blog> mainList();

Blog getBlogById(int blogId);

boolean addBlog(Blog blog);

boolean updateBlog(Blog blog);

boolean deleteBlog(Blog blog);

}

* 1. Create an Interface named IBlogCommentsDao.java

package com.coll.OnlineCollaborate.dao;

import java.util.List;

import com.coll.OnlineCollaborate.model.BlogComments;

public interface IBlogCommentsDao {

List<BlogComments> getAllBlogComments();

BlogComments getBlogCommentsById(int blogComemntId);

boolean addBlogComments(BlogComments blogComments);

boolean updateBlogComments(BlogComments blogComments);

boolean deleteBlogComments(BlogComments blogComments);

}

1. Create the DAO interface implementation classes inside com.coll.OnlineCollaborate.daoImpl package:
   1. Create a class named UserDaoImpl.java inside the above said package:

package com.coll.OnlineCollaborate.daoImpl;

import java.util.List;

import org.hibernate.query.Query;

import org.hibernate.SessionFactory;

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

import org.springframework.stereotype.Repository;

import org.springframework.transaction.annotation.Transactional;

import com.coll.OnlineCollaborate.dao.IUserDao;

import com.coll.OnlineCollaborate.model.User;

@Repository("userDao")

@Transactional

public class UserDaoImpl implements IUserDao{

@Autowired

SessionFactory sessionFactory;

@Override

public List<User> userListbyStatus(String status) {

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

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

return query.getResultList();

}

@Override

public List<User> getAllUsers() {

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

}

@Override

public User getUserById(int userId) {

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

}

@Override

public User getUserByUsername(String username) {

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

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

}

@Override

public User validateUser(User user) {

String username=user.getUsername();

String password=user.getPassword();

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

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

try {

user=(User)query.getSingleResult();

return user;

}

catch(Exception e) {

e.printStackTrace();

return null;

}

}

@Override

public boolean addUser(User user) {

try {

sessionFactory.getCurrentSession().save(user);

return true;

}

catch(Exception ex) {

ex.printStackTrace();

return false;

}

}

@Override

public boolean updateUser(User user) {

try {

sessionFactory.getCurrentSession().update(user);

return true;

}

catch(Exception ex) {

ex.printStackTrace();

return false;

}

}

@Override

public boolean deleteUser(int userId) {

try {

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

return true;

}

catch(Exception ex) {

ex.printStackTrace();

return false;

}

}

@Override

public boolean deactiveUser(int userId) {

try {

User user=getUserById(userId);

user.setEnabled(false);

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;

}

}

}

* 1. Create a class named BlogDaoImpl.java inside the above said package:

//Complete this class code

* 1. Create a class named BlogCommentsDaoImpl.java inside the above said package:

//Complete this class code

1. Create the service interfaces inside com.coll.OnlineCollaborate.service package:
   1. Create an interface named IUserService.java inside the above said package:

package com.coll.OnlineCollaborate.service;

import java.util.List;

import com.coll.OnlineCollaborate.model.User;

public interface IUserService {

List<User> userListbyStatus(String status);

List<User> getAllUsers();

User getUserById(int userId);

User getUserByUsername(String username);

User validateUser(User user);

boolean addUser(User user);

boolean updateUser(User user);

boolean deleteUser(int userId);

boolean deactiveUser(int userId);

boolean updateUserProfile(String file, Integer userId);

}

* 1. Create an interface named IBlogService.java inside the above said package:

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);

List<Blog> mainList();

Blog getBlogById(int blogId);

boolean addBlog(Blog blog);

boolean updateBlog(Blog blog);

boolean deleteBlog(Blog blog);

}

* 1. Create an interface named IBlogCommentsService.java inside the above said package:

package com.coll.OnlineCollaborate.service;

import java.util.List;

import com.coll.OnlineCollaborate.model.BlogComments;

public interface IBlogCommentsService {

List<BlogComments> getAllBlogComments();

BlogComments getBlogCommentsById(int blogComemntId);

boolean addBlogComments(BlogComments blogComments);

boolean updateBlogComments(BlogComments blogComments);

boolean deleteBlogComments(BlogComments blogComments);

}

1. Create the service implementation classes inside com.coll.OnlineCollaborate.serviceImpl package:
   1. Create a class named UserServiceImpl.java inside the above said package:

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 addUser(User user) {

return userDao.addUser(user);

}

@Override

public boolean updateUser(User user) {

return userDao.updateUser(user);

}

@Override

public boolean deleteUser(int userId) {

return userDao.deleteUser(userId);

}

@Override

public boolean deactiveUser(int userId) {

return userDao.deactiveUser(userId);

}

@Override

public boolean updateUserProfile(String file, Integer userId) {

return userDao.updateUserProfile(file, userId);

}

}

* 1. Create a class named BlogServiceImpl.java inside the above said package:

package com.coll.OnlineCollaboration.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.OnlineCollaboration.dao.IBlogDao;

import com.coll.OnlineCollaboration.model.Blog;

import com.coll.OnlineCollaboration.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 List<Blog> mainList() {

return blogDao.mainList();

}

@Override

public Blog getBlogById(int blogId) {

return blogDao.getBlogById(blogId);

}

@Override

public boolean addBlog(Blog blog) {

return blogDao.addBlog(blog);

}

@Override

public boolean updateBlog(Blog blog) {

return blogDao.updateBlog(blog);

}

@Override

public boolean deleteBlog(Blog blog) {

return blogDao.deleteBlog(blog);

}

}

* 1. Create a class named BlogCommentsServiceImpl.java inside the above said package:

package com.coll.OnlineCollaboration.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.OnlineCollaboration.dao.IBlogCommentsDao;

import com.coll.OnlineCollaboration.model.BlogComments;

import com.coll.OnlineCollaboration.service.IBlogCommentsService;

@Service

@Transactional

public class BlogCommentsServiceImpl implements IBlogCommentsService{

@Autowired

IBlogCommentsService blogCommentsService;

@Override

public List<BlogComments> getAllBlogComments() {

return blogCommentsService.getAllBlogComments();

}

@Override

public BlogComments getBlogCommentsById(int blogCommentId) {

return blogCommentsService.getBlogCommentsById(blogCommentId);

}

@Override

public boolean addBlogComments(BlogComments blogComments) {

return blogCommentsService.addBlogComments(blogComments);

}

@Override

public boolean updateBlogComments(BlogComments blogComments) {

return blogCommentsService.updateBlogComments(blogComments);

}

@Override

public boolean deleteBlogComments(BlogComments blogComments) {

return blogCommentsService.deleteBlogComments(blogComments);

}

}

1. Create the controller classes inside com.coll.OnlineCollaborate.controller package:
   1. Create a class named UserController.java inside the above package:

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.RequestMethod;

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

import com.coll.OnlineCollaborate.model.User;

import com.coll.OnlineCollaborate.service.IUserService;

@RestController

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

@RequestMapping(value="/api")

public class UserController {

@Autowired

IUserService userService;

@PostMapping("save-user")

public boolean saveUser(@RequestBody User user) {

return userService.addUser(user);

}

@GetMapping("user-list")

public List<User> allUsers() {

return userService.getAllUsers();

}

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

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

return userService.deleteUser(userId);

}

@GetMapping("user/{userId}")

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

return userService.getUserById(userId);

}

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

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

user.setUserId(userId);

return userService.updateUser(user);

}

@RequestMapping(value="login/{username,password}", method=RequestMethod.POST)

public User validateUser(@RequestBody User user,@PathVariable("username") String username, @PathVariable("password") String password) {

user.setUsername(username);

user.setPassword(password);

return userService.validateUser(user);

}

}

* 1. Create a class named BlogController.java inside the above package:

package com.coll.OnlineCollaboration.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.OnlineCollaboration.model.Blog;

import com.coll.OnlineCollaboration.service.IBlogService;

@RestController

@CrossOrigin(origins="http://localhost:8080")

@RequestMapping(value="/api")

public class BlogController {

@Autowired

IBlogService blogService;

@PostMapping("save-blog")

public boolean saveBlog(@RequestBody Blog blog)

{

return blogService.addBlog(blog);

}

@GetMapping("blog-list")

public List<Blog> allBlog()

{

return blogService.getAllBlogs();

}

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

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

{

return blogService.deleteBlog(blogId);

}

@GetMapping("blog/{blogId}")

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

{

return blogService.getBlogById(blogId);

}

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

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

{

blog.setBlogId(blogId);

return blogService.updateBlog(blog);

}

}

* 1. Create a class named BlogCommentsrController.java inside the above package:

package com.coll.OnlineCollaboration.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.OnlineCollaboration.model.BlogComments;

import com.coll.OnlineCollaboration.service.IBlogCommentsService;

@RestController

@CrossOrigin(origins="http://localhost:8080")

@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/{blogCommentId}")

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

{

return blogCommentsService.deleteBlogComments(blogCommentId);

}

@GetMapping("blogComments/{blogCommentId}")

public BlogComments blogCommentById(@PathVariable("blogCommentId") int blogCommentId)

{

return blogCommentsService.getBlogCommentsById(blogCommentId);

}

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

public boolean updateBlogComments(@RequestBody BlogComments blogComments,@PathVariable("blogCommentId") int blogCommentId)

{

blogComments.setBlogCommentId(blogCommentId);

return blogCommentsService.updateBlogComments(blogComments);

}

}

1. Save All
2. Right click on OnlineCollaborateApplication .java and run the project.

FRONTEND

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

ng new OnlineCollaborationAngular

2. Install Bootstrap CSS framework. Use the following command to install bootstrap in the project:

npm install bootstrap@3.3.7 --save

3. Include the following code in the styles.scss file:

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

4. Install Angular datatables. Use the following command to install angular-datatables in the project:

npm install angular-datatables --save

5. Open app.module.ts file and include required modules:

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

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

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

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

import {DataTablesModule} from 'angular-datatables';

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

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

@NgModule({

declarations: [

AppComponent

],

imports: [

BrowserModule,

AppRoutingModule,

FormsModule,

DataTablesModule,

HttpClientModule,

ReactiveFormsModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

6. Generate following Components:

a. register-user:

ng g c components/register-user

b. user-list:

ng g c components/user-list

c. login-user:

ng g c components/login-user

d. activate-user:

ng g c components/activate-user

e. add-blog:

ng g c components/add-blog

f. approve-blog:

ng g c components/approve-blog

g. view-blog:

ng g c components/view-blog

7. Create models classes:

a. user:

ng g class model/user

b. blog:

ng g class model/blog

8. Create services:

a. user

ng g s services/user

b. blog:

ng g s services/blog

8. Open app.module.ts and register service classes:

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

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

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

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

import {DataTablesModule} from 'angular-datatables';

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

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

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

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

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

import { ActivateUserComponent } from './components/activate-user/activate-user.component';

import { AddBlogComponent } from './components/add-blog/add-blog.component';

import { ApproveBlogComponent } from './components/approve-blog/approve-blog.component';

import { ViewBlogComponent } from './components/view-blog/view-blog.component';

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

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

@NgModule({

declarations: [

AppComponent,

RegisterUserComponent,

UserListComponent,

LoginUserComponent,

ActivateUserComponent,

AddBlogComponent,

ApproveBlogComponent,

ViewBlogComponent

],

imports: [

BrowserModule,

AppRoutingModule,

FormsModule,

DataTablesModule,

HttpClientModule,

ReactiveFormsModule

],

providers: [UserService, BlogService],

bootstrap: [AppComponent]

})

export class AppModule { }

9. Open app-routing.module.ts file and edit as following:

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

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

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

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

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

import { ActivateUserComponent } from './components/activate-user/activate-user.component';

import { AddBlogComponent } from './components/add-blog/add-blog.component';

import { ApproveBlogComponent } from './components/approve-blog/approve-blog.component';

import { ViewBlogComponent } from './components/view-blog/view-blog.component';

const routes: Routes = [

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

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

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

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

{path:'activate-user', component:ActivateUserComponent},

{path:'add-blog', component:AddBlogComponent},

{path:'approve-blog', component:ApproveBlogComponent},

{path:'view-blog', component:ViewBlogComponent},

];

@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule]

})

export class AppRoutingModule { }

10. Open app.component.html(landing page) file and edit as following:

<div class="container-fluid">

<nav class="navbar navbar-expand-sm bg-dark navbar-dark">

<ul class="navbar-nav">

<li class="nav-item">

<a routerLink="user-list" class="nav-link btn btn-primary" role="button">View Users</a>

</li>

<li class="nav-item">

<a routerLink="register-user" class="nav-link btn btn-primary" role="button">Registration</a>

</li>

<li class="nav-item">

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

</li>

</ul>

</nav>

<router-outlet></router-outlet>

</div>

11. Open user.ts file(model class) and define the class:

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;

}

12. Open user.service.ts file and define the service class:

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);

}

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

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

}

}

13. Opened register-user.component.ts file. and edited as following:

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

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

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

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

@Component({

selector: 'app-register-user',

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

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

})

export class RegisterUserComponent implements OnInit {

user:User=new User();

submitted=false;

constructor(private userservice:UserService) { }

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(register){

this.user=new User();

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

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

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

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

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

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

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

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

this.user.enabled=true;

this.user.status="Active";

}

else{

this.user.enabled=false;

this.user.status="Inactive";

}

this.user.isOnline=false;

this.submitted=true;

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();

}

}

13. Open register-user.component.html file and modify as folloiwng:

<h3>Register Here</h3>

<div class="row">

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

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

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

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

<div class="form-group">

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

<input type="text" formControlName="firstName" class="form-control" data-toggle="tooltip" data-placement="right" title="Enter First Name">

<div class="alert alert-danger" \*ngIf="(FirstName.touched) && (FirstName.invalid)" style="margin-top: 5px;">

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

</div>

</div>

<div class="form-group">

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

<input type="text" formControlName="lastName" class="form-control" data-toggle="tooltip" data-placement="right" title="Enter Last Name">

<div class="alert alert-danger" \*ngIf="(LastName.touched) && (LastName.invalid)" style="margin-top: 5px;">

<span \*ngIf="LastName.error.required">Last Name is required</span>

</div>

</div>

<div class="form-group">

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

<input type="text" formControlName="username" class="form-control" data-toggle="tooltip" data-placement="right" title="Enter Username">

<div class="alert alert-danger" \*ngIf="(Username.touched) && (Username.invalid)" style="margin-top: 5px;">

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

</div>

</div>

<div class="form-group">

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

<input type="text" formControlName="password" class="form-control" data-toggle="tooltip" data-placement="right" title="Enter Password">

<div class="alert alert-danger" \*ngIf="(Password.touched) && (Password.invalid)" style="margin-top: 5px;">

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

</div>

</div>

<div class="form-group">

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

<input type="text" formControlName="confirm\_password" class="form-control" data-toggle="tooltip" data-placement="right" title="Confirm Password" pattern="{{Password.value}}">

<div class="alert alert-danger" \*ngIf="(ConfirmPassword.touched) && (ConfirmPassword.invalid)" style="margin-top: 5px;">

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

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

</div>

</div>

<div class="form-group">

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

<input type="text" formControlName="email" class="form-control" data-toggle="tooltip" data-placement="right" title="Enter Email Id">

<div class="alert alert-danger" \*ngIf="(Email.touched) && (Email.invalid)" style="margin-top: 5px;">

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

<span \*ngIf="Email.error.email">Invalid Email format</span>

</div>

</div>

<div class="form-group">

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

<select formControlName="role" class="form-control" data-toggle="tooltip" data-placement="right" title="Select user Role">

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

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

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

</select>

</div>

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

</form>

</div>

</div>

<div>

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

</div>

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

<div [hidden]="!submitted">

<h4>Congratulations! You have registered successfully!</h4>

</div>

</div>

</div>

14. Open user-list.comment.ts and modify as following:

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

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

import {User} from '../../model/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 {

usersArray:any=[];

dtOptions: DataTables.Settings={};

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

users: Observable<User[]>;

user: User=new User();

deleteMessage=false;

userlist:any;

isupdated=false;

constructor(private userservice: UserService) { }

ngOnInit(): void {

this.isupdated=false;

this.dtOptions={

pageLength:6,

stateSave:true,

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

processing:true

};

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

this.users=data;

this.dtTrigger.next();

})

}

deleteUser(id:number){

this.userservice.deleteUser(id)

.subscribe(data=>{

console.log(data);

this.deleteMessage=true;

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

this.users=data;

})

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

}

updateUser(id:number){

this.userservice.getUser(id)

.subscribe(data=>{

this.userlist=data;

}, error=>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()

});

update(user){

this.user=new User();

this.user.userId=this.UserId.value,

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

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

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

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

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

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

this.user.enabled=true;

this.user.status="Active";

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 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');

}

changeidUpdate(){

this.isupdated=false;

}

}

.**Save All.**

.**Run the SpringBoot Application**

.**Run the Angular project**

**Login-User.ts**

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

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

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

import { Observable, Subject } from 'rxjs';

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

import { DataTablesModule } from 'angular-datatables';

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

@Component({

  selector: 'app-login-user',

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

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

})

export class LoginUserComponent implements OnInit {

  user : User=new User();

  currentUser : any;

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

  ngOnInit(): void {

  }

  loginform=new FormGroup({

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

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

  })

  validateUser() {

    this.user=new User();

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

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

    this.userService.createUser(this.user).subscribe (

      data => {

        console.log(data);

        if(data!=null) {

          this.currentUser=data;

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

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

         }

         else{

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

         }

        }

        else {

          console.log("Object Empty");

        }

      },

      error => console.log(error)

    )

  }

  get Username() {

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

  }

  get Password() {

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

  }

  RegUser(){

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

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

  }

}

**Login user-component.html**

<!DOCTYPE html>

<html>

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

<body>

    <div class="login-dark">

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

            <h2 class="sr-only">Login Form</h2>

            <div class="illustration"><i class="icon ion-ios-locked-outline"></i></div>

      <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>

      <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 class="form-group"><button class="btn btn-primary btn-block" type="submit">Log In</button></div>

          </form>

    </div>

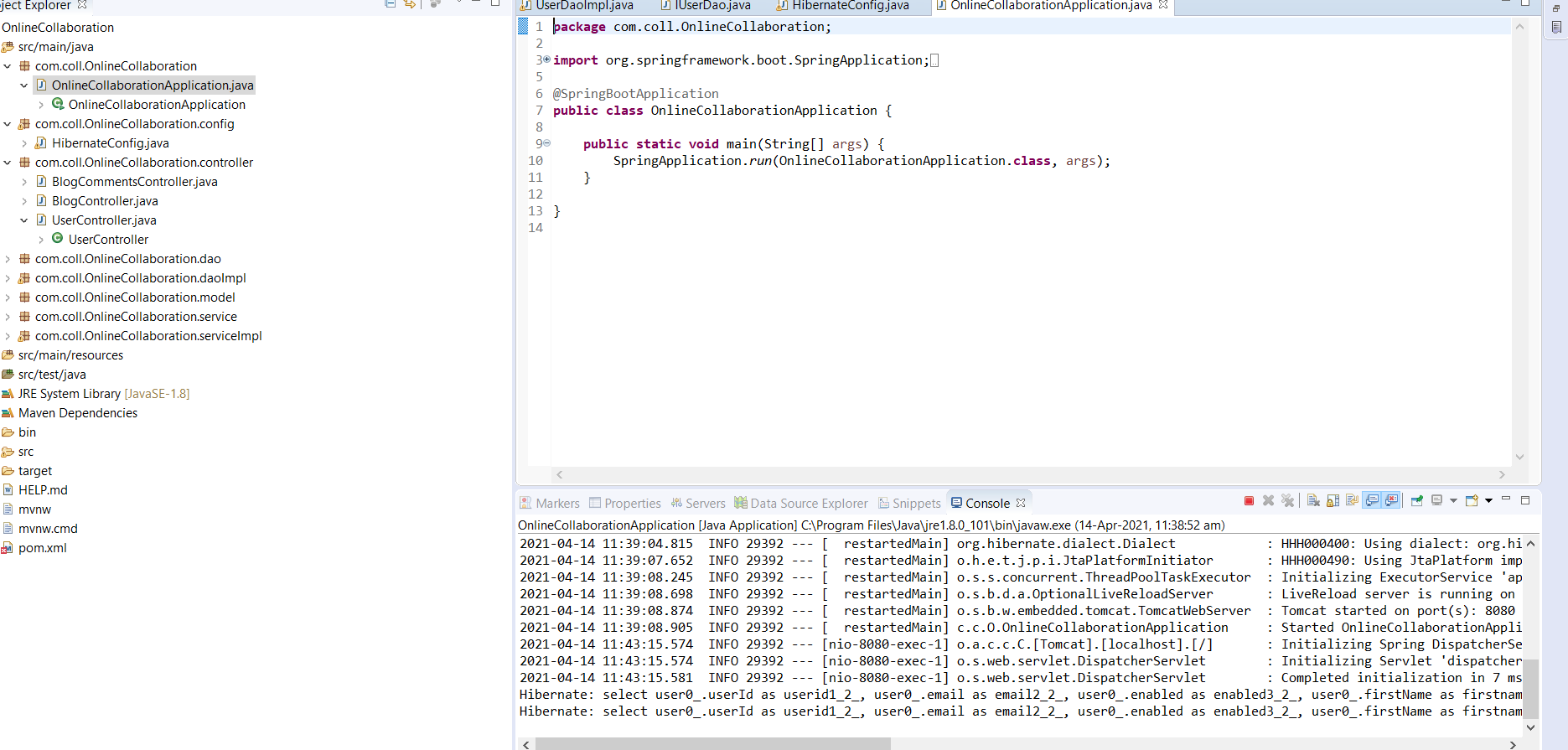
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>

    <script src="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.1.3/js/bootstrap.bundle.min.js"></script>

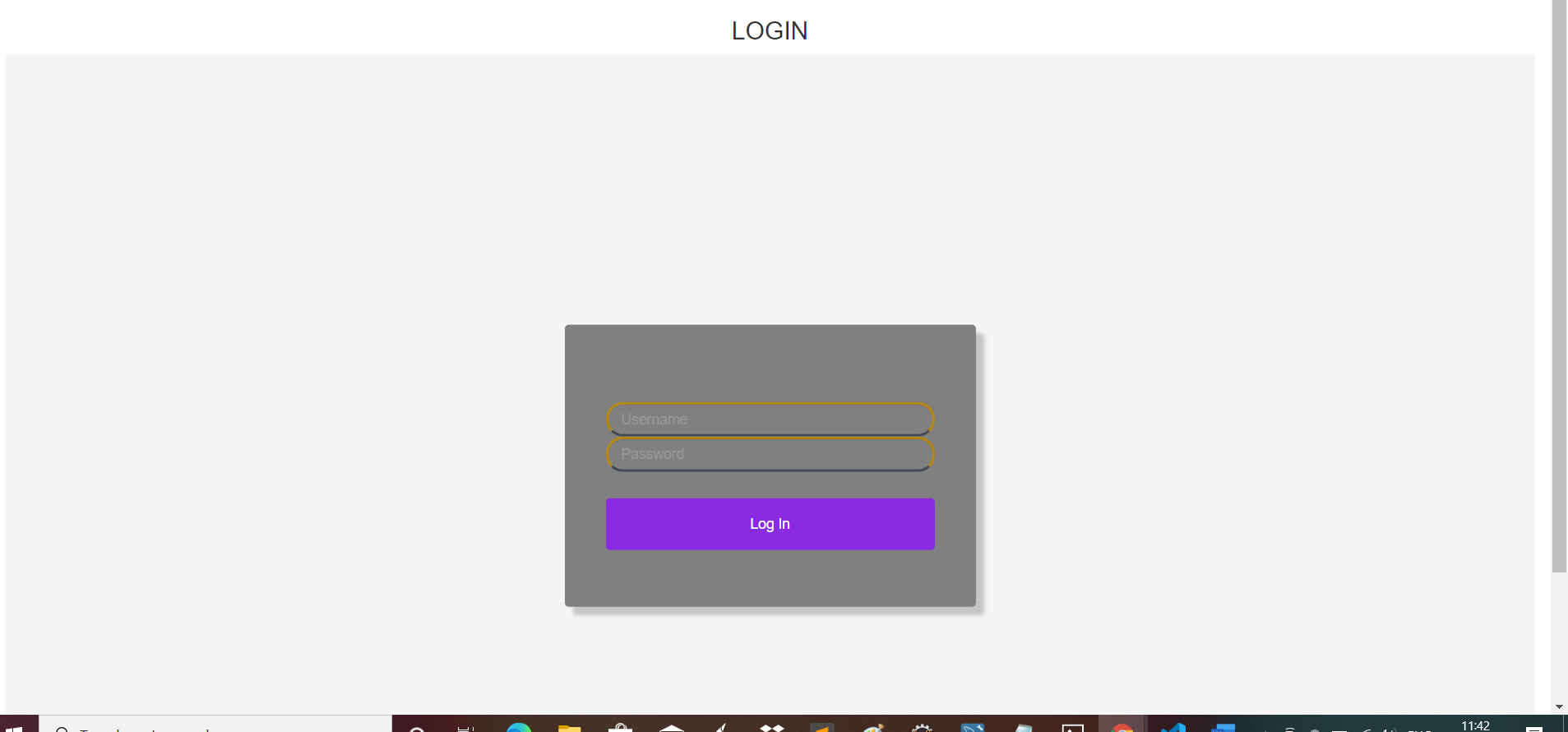
</body>

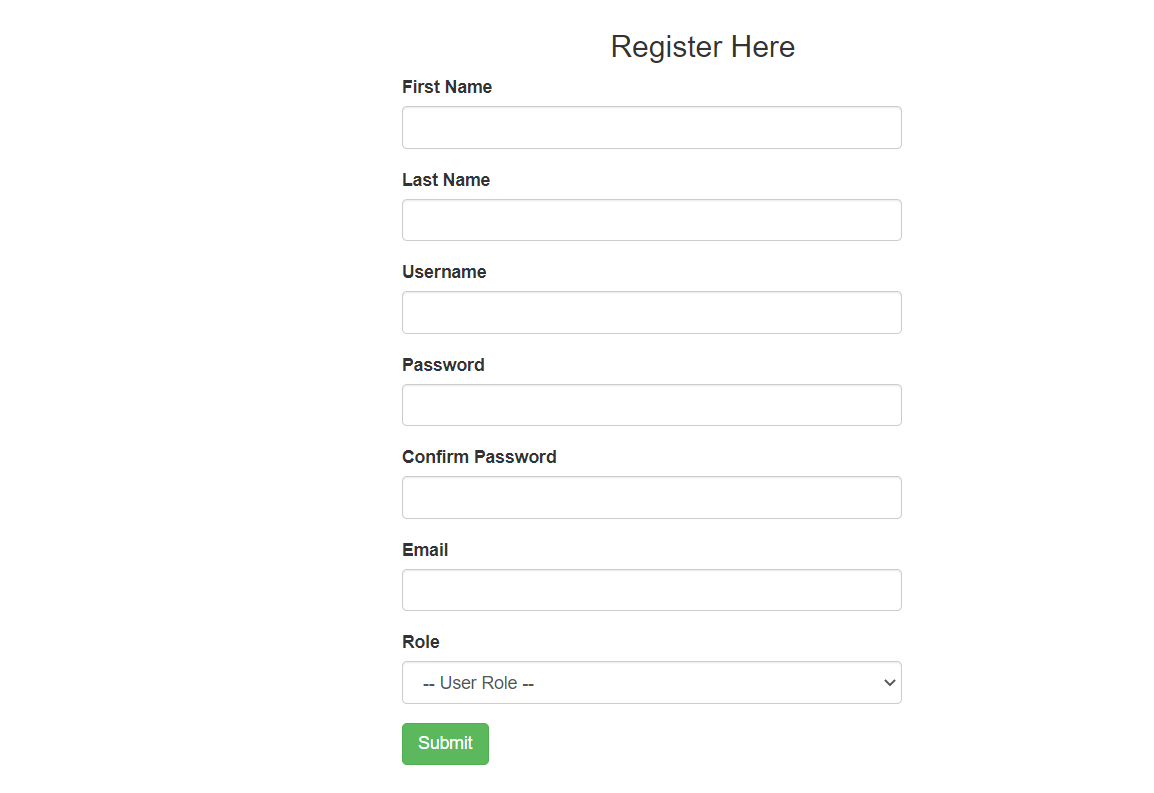
</html>

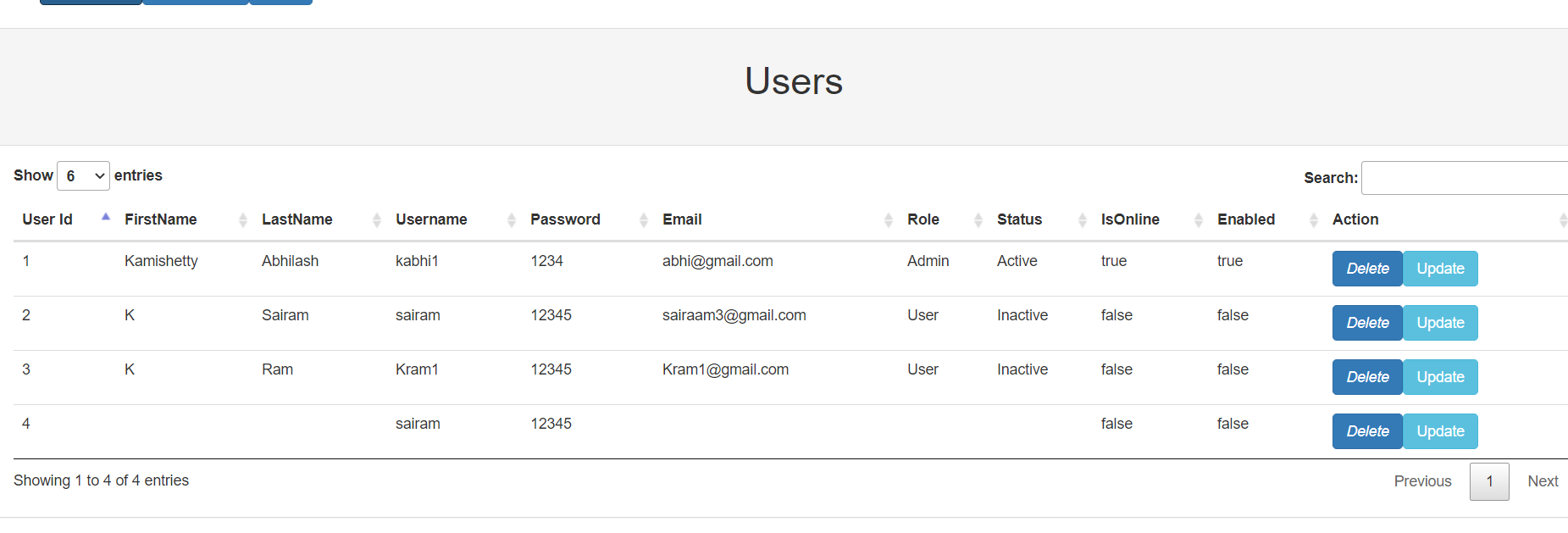
Run OnlineCollaboration.Java



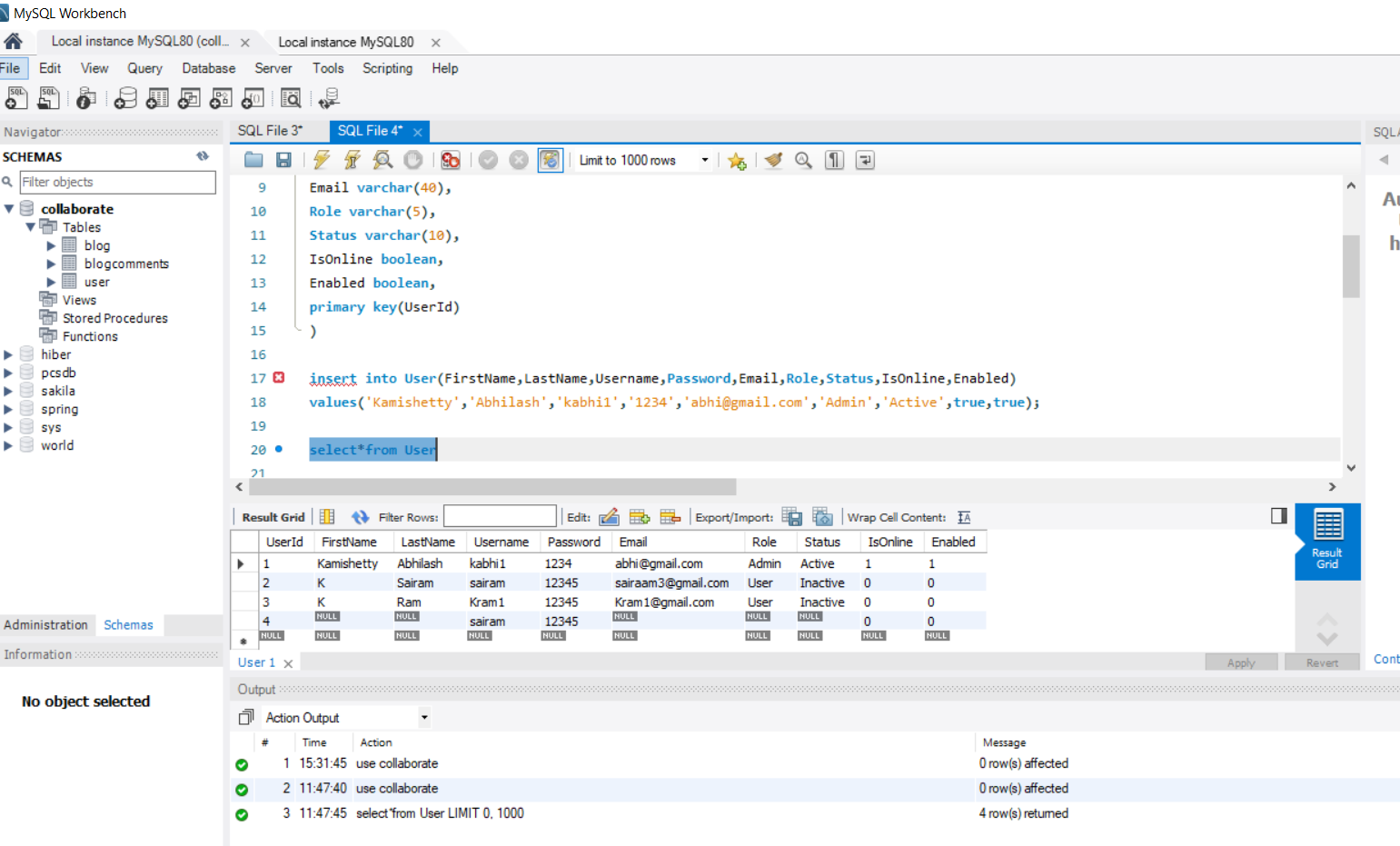
Login







Database



CONCLUSION

The Online Collaborate project helps to connect with people.

FEATURE WORKS

* Job Posting
* Forum
* BlogComments
* Friend

**REFERENCES**

NIIT Project 1 Book

www.Github.com