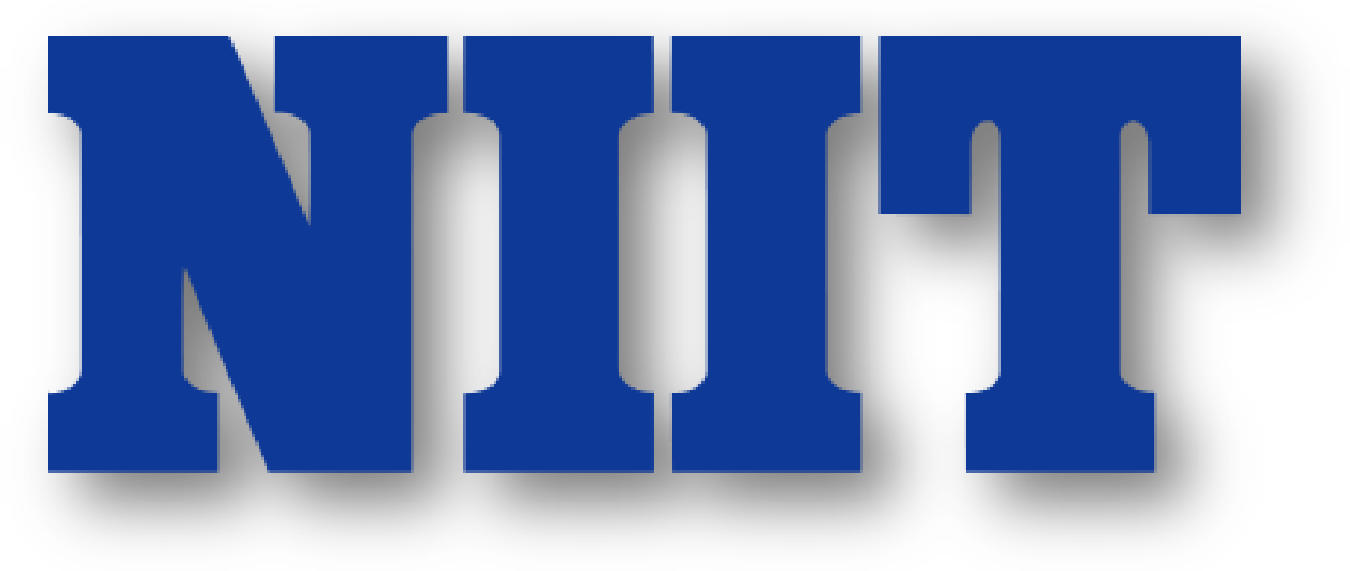
PROJECT ON

**Professionet Consultancy Services (PCS) Application**

DEVELOPED By –

Bheema Yugandhar Reddy



**Professionet Consultancy Services (PCS) Application**

Batch Code : S210192

Start Date :

End Date :

Name of the

Coordinator : Mrs. Lopamudra Bera

Name of the

Developer : Bheema

Yugandhar Reddy

Date of

Submission :

**Professionet Consultancy Services (PCS) Application**

CERTIFICATE

This is to certify that this report, titled Online collaboration embodies the original work done by Bheema Yugandhar Reddy, in partial fulfillment of their course requirement at NIIT.

CO-ORDINATOR:

Acknowledgement

*We have benefitted a lot from the feedback and suggestions given to us by Mrs. Lopamudra Bera and other faculty members.*

Abstract

Online Collaboration is the application developed for blogging purpose. In this application their will be one admin and users. Admin will check the blogs posted by users after verifying the content in their blogs admin will give access to the blogs. After getting access anyone can view the content of the blog in Online collaboration application

Configuration

Hardware

Processor : Intel i3 or higher.

RAM : 2GB(minimum)

Speed : 1.5GHz

Secondary

Storage : 10GB

Software :

Java Version : JDK 1.8

Database

Management: MySQL

IDE : Visual Studio code

OS : Windows 10.

INDEX

|  |  |  |
| --- | --- | --- |
| SNO | Title | Page No. |
| 1 | Introduction | 08 |
| 2 | Aim and Objective | 09 |
| 3 | Case Study | 10 |
| 4 | Project Requirements and Specification | 12 |
| 5 | Project Analysis | 13 |
| 6 | Implementation and Diagrams | 14 |
| 7 | Source Code Snippets | 17 |
| 8 | Outputs | 34 |
| 9 | Challenges Faced | 36 |
| 10 | References | 37 |

Introduction

This project will help you to implement all the concepts. Strategies, techniques, and technologies its that you are laming in the current ten. In this project. you will integrate all those concepts and mate live project by following all the aspects of software engineering and project management. You will implement various software engineering and project management concepts in each and every phase of project development. In addition. this project makes you go through the entire software project life cycle, which will help you to work in a real time application development environment.

[1]Aspiring professionals need to apply their knowledge. skills and concepts that they have learned to develop a software project. In real Inv environment. software companies expect their employs to develop their software applications, which incorporate all the pluses of software engineering and project management. This project enables you to work in a simulated environment where you is ill work as a tram and develop a software application. This project provides a detailed study and implementation of project planning. analysts design. development. and documentation. After completing this project, you will get a confidence to apply your experience while developing projects in a software industry. You will appreciate and understand the need and usage of all the theoretical concepts that you have teamed in various semesters.

AIM and Objective

The aim of this project is to design and develop an application for a blogging application named as Online Collaboration

Objective :

* To connect to the data base.
* Let the admin add and approve and maintain the blogs and the records.
* Provide a good user interface.
* Test the application.

Case Study

Project Requirement Specification

For Employers/Project Managers:

■ Create an administrator profile.

■ Create users profile.

■ Authenticate admin profile by UserId and password entered

■ Create blogs and posting in the web.

■ Autherise the blogs by the admin.

Project Analysis

Implementation and Diagrams

[3]Java has been one of the most popular programming language for many years. Java is Object Oriented. However it is not considered as pure object oriented as it provides support for primitive data types (like int, char, etc).The Java codes are first compiled into byte code (machine independent code). Then the byte code is run on **J**ava **V**irtual **M**achine (JVM) regardless of the underlying architecture. Java syntax is similar to C/C++. But Java does not provide low level programming functionalities like pointers. Also, Java codes are always written in the form of classes and objects.

Java is used in all kind of applications like Mobile Applications (Android isJava based), desktop applications, web applications, client server applications, enterprise applications and many more.

Angular applications are built using TypeScript language, a superscript for JavaScript, which ensures higher security as it supports types (primitives, interfaces, etc.). It helps catch and eliminate errors early when writing the code or performing maintenance tasks.

Unlike CoffeeScript or Dart, TypeScript is not a stand-alone language. With TypeScript, you can easily take the existing ES5 or ES2015+ JS code and it will.

You can directly debug TypeScript code in the browser or an editor if you have proper map files created during build time. This language ensures improved navigation, refactoring, and autocompletion services. You can even opt out of its inbuilt features when needed.

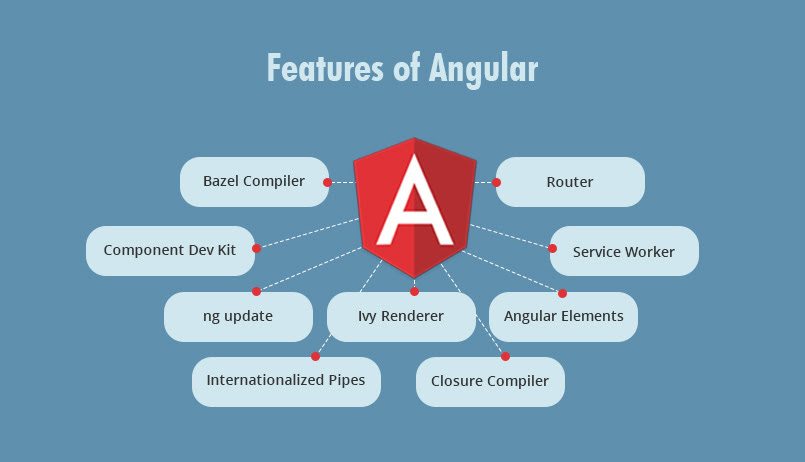
### Declarative UI

Angular uses HTML to define the UI of the application. HTML, as compared to JavaScript, is a less convoluted language. HTML is also a declarative and intuitive language.

How does it help? You don’t need to invest your time in program flows and in deciding what loads first. Define what you require and Angular will take care of it.

### POJO

With Angular, you don’t need any additional getter and setter functions. Since, every object it uses is POJO (Plain Old JavaScript Object), which enables object manipulation by providing all the conventional JavaScript functionalities. You can remove or add properties from the objects, while also looping over these objects when required.



Some Code Snippets

Database:

1. Open MySQL Workbench
2. Create a schema named collaborate
3. Create initially following tables:

* Active the Schema

use collaboration;

* Create User table

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)

)

* Insert one record with Admin role into User table:
* View the record:

select \* from User

* Create Blog table

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)

)

* Create BlogComments table

create table BlogComments(

BlogCommentId int not null auto\_increment,

UserId int,

Username varchar(20),

UserProfileId varchar(20),

Title varchar(30),

NonOfLikes int,

BlogComment varchar(50),

CurrentDate Date,

BlogId int,

primary key(BlogCommentId)

)

Project: Backend

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;

//Generate getter and setter methods

}

* 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;

//Generate getter and setter methods

}

* 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;

//Generate getter and setter methods

}

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:

//Complete code

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

//Complete code

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:

//Complete code

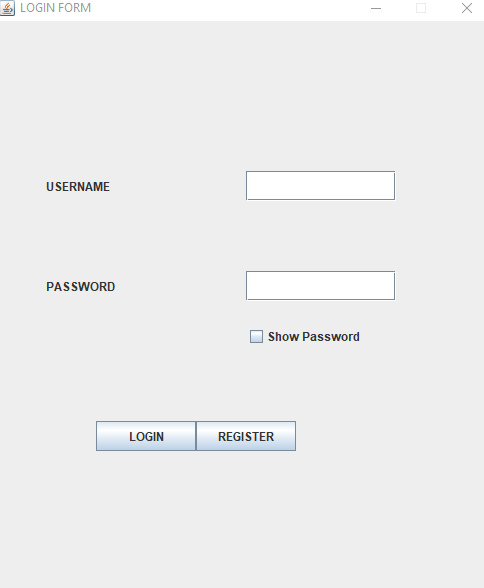
* 1. Create a class named BlogCommentsrController.java inside the above package:

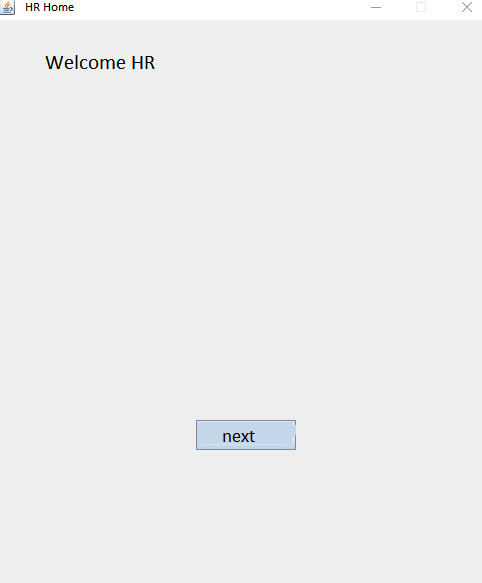
//Complete code

1. Save All
2. Right click on OnlineCollaborateApplication.java class and Run as Java Application.

Outputs

Login :





Challenges Faced

* The initial steps to connect the MySQL to the Eclipse together.
* Sometimes retrieval of large data was taking too much time.
* When open in multiple frames some pre-existing frames were not closing or hiding.

References

[1] NIIT Project Guide

[2] NIIT Lab Guide

[3] Wikipedia

[4] Google Images