

**Online Collaboration**

Batch Code : S210192

Start Date : 24th February 2021

End Date : 9th April 2021

Name of The Coordinator : Mrs.Lopamudra Bera

Name of The Developer : Ravula Saipreethi

Date of Submission : 14h April

**ABSTRACT**

This Project is an application for the OnlineCollaboration that provides the client to freely be able to manage, store and maintain their company records with a good user interface.

The primary goal of this project is to demonstrate the new technologies Angular JS,RESTful Web Services,and the database.I

**ACKNOWLEDGEMENT**

It is My great pleasure to thanks of gratitude to my teacher Mrs.Lopamudra Bera for valuable guidance through provoking discussions, vital suggestions and sharing valuable expertise throughout the project on the topic Online Collaboration.

**CONFIGURATION**

**Hardware**

Processor : Intel i3 or higher

RAM : 2GB (minimum)

Speed Secondary : 1.5GHz

Storage : 10GB

**Software**

Java Version : JDK 1.8

Database Management : MYSQL

IDE : Eclipse

Visual studio : 11.2

Os : Windows 10

**INDEX**

**DESCRIPTION Page No**

Aim and objective of the project 7

Implementation 8

Coding 13

Output 49

References 51

**AIM AND OBJECTIVE OF THE PROJECT**

**Aim:**

The aim of the project is to design and develop a application for the OnlineCollabration.

**Objective:**

To connect the data base.

Let the admin add update modify and maintain the database and the records.

Provide a good user interface.

Test the application.

**IMPLEMENTATION**

Java has been one of the most popular programming languages 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 is Java based), desktop applications, web applications, client server applications, enterprise applications and many more.

When compared with C++, Java codes are generally more maintainable because Java does not allow many things which may lead bad/inefficient programming if used incorrectly. For example, non-primitives are always references in Java. So, we cannot pass large objects (like we can do in C++) to functions, we always pass references in Java. One more example, since there are no pointers, bad memory access is also not possible.

Each UI component has its place on the interface. The location of a component is determined by the class used to layout the components.

So far, we have covered the basic programming constructs (such as variables, data types, decision, loop, array and method) and introduced the important concept of Object-Oriented Programming (OOP). As discussed, OOP permits higher level of abstraction than traditional Procedural-Oriented languages (such as C and Pascal). You can create high-level abstract data types called *classes* to mimic real-life things. These classes are self-contained and are *reusable*.

In this article, I shall show you how you can *reuse* the graphics classes provided in JDK for constructing your own Graphical User Interface (GUI) applications. Writing your own graphics classes (and re-inventing the wheels) is mission impossible! These graphics classes, developed by expert programmers, are highly complex and involve many advanced *design patterns*.  However, re-using them are not so difficult, if you follow the API documentation, samples and templates provided.

* Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages (such as C++, C#, Java, Python, PHP, Go) and runtimes (such as .NET and Unity).

Visual Studio Code is a source-code editor that can be used with a variety of programminglanguages,including [Java](https://en.wikipedia.org/wiki/Java_(programming_language)" \o "Java (programming language)), [JavaScript](https://en.wikipedia.org/wiki/JavaScript" \o "JavaScript), [Go](https://en.wikipedia.org/wiki/Go_(programming_language)" \o "Go (programming language)), [Node.js](https://en.wikipedia.org/wiki/Node.js" \o "Node.js), [Python](https://en.wikipedia.org/wiki/Python_(programming_language)" \o "Python (programming language)) and [C++](https://en.wikipedia.org/wiki/C++" \o "C++).It is based on the [Electron](https://en.wikipedia.org/wiki/Electron_(software_framework)" \o "Electron (software framework)) framework,which is used to develop [Node.js](https://en.wikipedia.org/wiki/Node.js" \o "Node.js) [Web applications](https://en.wikipedia.org/wiki/Web_application" \o "Web application) that run on the [Blink layout engine](https://en.wikipedia.org/wiki/Blink_layout_engine" \o "Blink layout engine). Visual Studio Code employs the same editor component (codenamed "Monaco") used in [Azure DevOps](https://en.wikipedia.org/wiki/Azure_DevOps_Server" \o "Azure DevOps Server) (formerly called Visual Studio Online and Visual Studio Team Services).

Instead of a project system, it allows users to open one or more directories, which can then be saved in workspaces for future reuse. This allows it to operate as a [language-agnostic](https://en.wikipedia.org/wiki/Language-agnostic" \o "Language-agnostic) code editor for any language. It supports a number of programming languages and a set of features that differs per language. Unwanted files and folders can be excluded from the project tree via the settings. Many Visual Studio Code features are not exposed through menus or the user interface but can be accessed via the command palette.

Visual Studio Code can be extended via [extensions](https://en.wikipedia.org/wiki/Plug-in_(computing)" \o "Plug-in (computing)), available through a central repository. This includes additions to the editor and language support. A notable feature is the ability to create extensions that add support for new [languages](https://en.wikipedia.org/wiki/Programming_language" \o "Programming language), [themes](https://en.wikipedia.org/wiki/Theme_(computing)" \o "Theme (computing)), and [debuggers](https://en.wikipedia.org/wiki/Debugger" \o "Debugger), perform [static code analysis](https://en.wikipedia.org/wiki/Static_code_analysis" \o "Static code analysis), and add [code linters](https://en.wikipedia.org/wiki/Lint_(software)" \o "Lint (software)) using the [Language Server Protocol](https://en.wikipedia.org/wiki/Language_Server_Protocol" \o "Language Server Protocol).

Visual Studio Code includes multiple extensions for [FTP](https://en.wikipedia.org/wiki/FTP" \o "FTP), allowing the software to be used as a free alternative for web development. Code can be synced between the editor and the server, without downloading any extra software.

Visual Studio Code allows users to set the [code page](https://en.wikipedia.org/wiki/Code_page" \o "Code page) in which the active document is saved, the [newline](https://en.wikipedia.org/wiki/Newline" \o "Newline) character, and the programming language of the active document. This allows it to be used on any platform, in any locale, and for any given programming language.

### **Language support:**

Out-of-the-box, Visual Studio Code includes basic support for most common programming languages. This basic support includes syntax highlighting, bracket matching, [code folding](https://en.wikipedia.org/wiki/Code_folding" \o "Code folding), and configurable snippets. Visual Studio Code also ships with [IntelliSense](https://en.wikipedia.org/wiki/Intelligent_code_completion" \o "Intelligent code completion) for JavaScript, TypeScript, JSON, CSS, and HTML, as well as debugging support for Node.js. Support for additional languages can be provided by freely available extensions on the VS Code Marketplace.

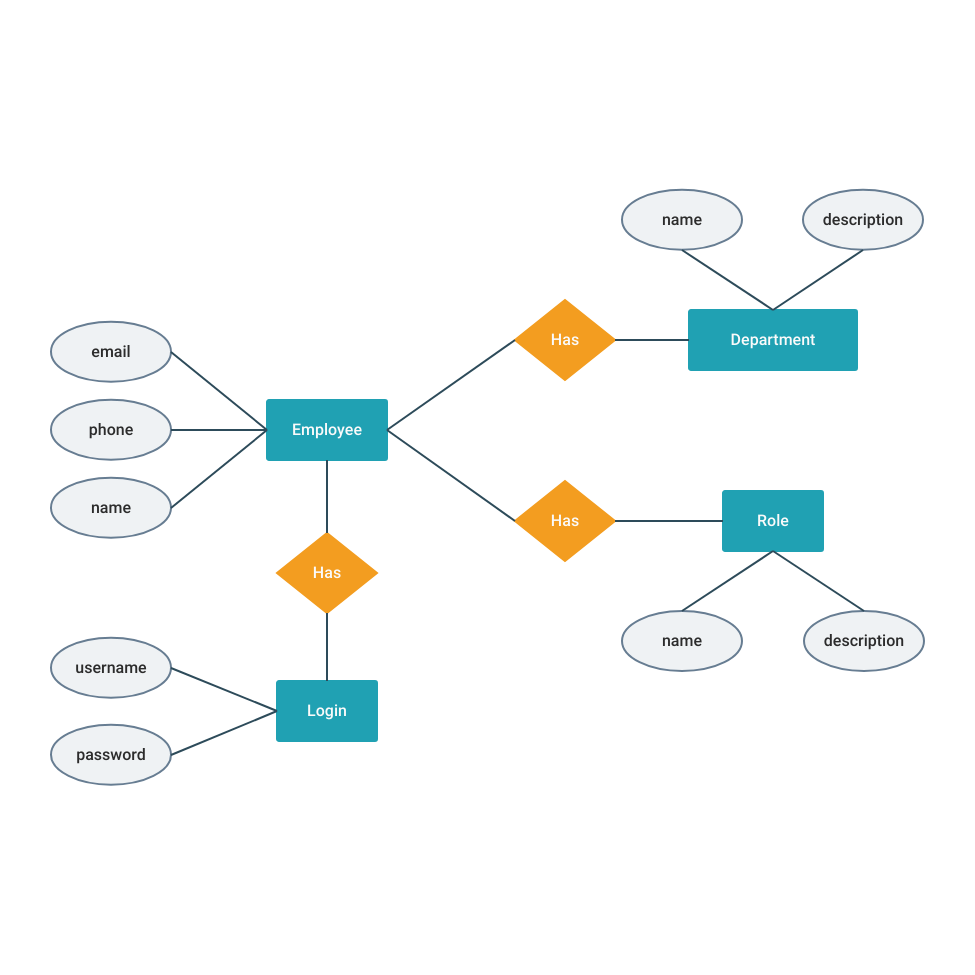
### **Data collection:**

Visual Studio Code [collects usage data and sends it to Microsoft](https://en.wikipedia.org/wiki/Telemetry" \l "Software" \o "Telemetry), although this can be disabled. In addition, because of the open-source nature of the application, the telemetry code is accessible to the public, who can see exactly what is collected. According to Microsoft, the data is shared with Microsoft-controlled affiliates and subsidiaries, although [law enforcement](https://en.wikipedia.org/wiki/Law_enforcement" \o "Law enforcement) may request it as part of a legal process.

### **Version control:**

[Source control](https://en.wikipedia.org/wiki/Source_control" \o "Source control) is a built-in feature of Visual Studio Code. It has a dedicated tab inside of the menu bar where you can access version control settings and view changes made to the current project. To use the feature you must link Visual Studio Code to any supported version control system (Git, Subversion, Perforce, etc.). This allows you to create repositories as well as make push and [pull requests](https://en.wikipedia.org/wiki/Pull_request" \o "Pull request) directly from the Visual Studio Code program.

**ER-DAIGRAM:**

****

**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)

)

**CODING:**

**Project: Backend:**

1. Create a SpringBoot project named “OnlineCollaboration”(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.OnlineCollaboration.config package and specify the required configuration in it. However, there is one more configuration class OnlineCollaborationApplication.java. This class is provided by Spring Boot automatically.

**package** com.coll.OnlineCollaboration.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/collaborate";

**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***="123456789";

@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;

}

}

4.Create Entity(Model) classes. Here, we are creating an Entity/POJO (Plain Old Java Object) class inside com.coll.OnlineCollaboration.model package.

a.Create a class inside the above said package named “User”.

**package** com.coll.OnlineCollaboration.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** isOnline() {

**return** isOnline;

}

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

**this**.isOnline = isOnline;

}

**public** **boolean** isEnabled() {

**return** enabled;

}

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

**this**.enabled = enabled;

}

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

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

}

}

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

**package** com.coll.OnlineCollaboration.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;

}

}

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

**package** com.coll.OnlineCollaboration.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***;

}

}

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

**package** com.coll.OnlineCollaboration.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***;

}

}

5.Create the DAO interfaces inside com.coll.OnlineCollaboration.dao package:

a.Create an Interface named IUserDao.java

**package** com.coll.OnlineCollaboration.dao;

**import** java.util.List;

**import** com.coll.OnlineCollaboration.model.User;

**public** **interface** IUserDao {

List<User> userListbyStatus(String status);

List<User> getAllUsers();

User getUserById(**int** userId);

User getUserByUsername(String username);

User validateUser(User user);

**boolean** addUser(User user);

**boolean** updateUser(User user);

**boolean** deleteUser(**int** userId);

**boolean** deactiveUser(**int** userId);

List<User> getAllInactiveUsers();

**boolean** activeUser(**int** userId);

**boolean** updateUserProfile(String file, Integer userId);

List<User> getAlldeactiveusers();

}

b.Create an Interface named IBlogDao.java

**package** com.coll.OnlineCollaboration.dao;

**import** java.util.List;

**import** com.coll.OnlineCollaboration.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);

}

c.Create an Interface named IBlogCommentsDao.java

**package** com.coll.OnlineCollaboration.dao;

**import** java.util.List;

**import** com.coll.OnlineCollaboration.model.BlogComments;

**public** **interface** IBlogCommentsDao {

List<BlogComments> getAllBlogComments();

BlogComments getBlogCommentsById(**int** blogCommentId);

**boolean** addBlogComments(BlogComments blogComments);

**boolean** updateBlogComments(BlogComments blogComments);

**boolean** deleteBlogComments(BlogComments blogComments);

}

6.Create the DAO interface implementation classes inside

com.coll.OnlineCollaboration.daoImpl package:

a.Create a class named UserDaoImpl.java inside the above said package:

**package** com.coll.OnlineCollaboration.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.OnlineCollaboration.dao.IUserDao;

**import** com.coll.OnlineCollaboration.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** List<User> getAllInactiveUsers() {

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

}

@Override

**public** **boolean** activeUser(**int** userId) {

**try** {

User user=getUserById(userId);

user.setEnabled(**true**);

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> getAlldeactiveusers() {

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

}

}

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

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

**import** com.coll.OnlineCollaboration.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) {

**return** (List<Blog>) sessionFactory.getCurrentSession().get(Blog.**class**, Integer.*valueOf*(id));

}

@Override

**public** List<Blog> mainList() {

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

}

@Override

**public** Blog getBlogById(**int** blogId) {

**return** sessionFactory.getCurrentSession().get(Blog.**class**, Integer.*valueOf*(blogId));

}

@Override

**public** **boolean** addBlog(Blog blog) {

**try** {

sessionFactory.getCurrentSession().save(blog);

**return** **true**;

}

**catch**(Exception ex) {

ex.printStackTrace();

**return** **false**;

}

}

@Override

**public** **boolean** updateBlog(Blog blog) {

**try** {

sessionFactory.getCurrentSession().update(blog);

**return** **true**;

}

**catch**(Exception ex) {

ex.printStackTrace();

**return** **false**;

}

}

@Override

**public** **boolean** deleteBlog(Blog blog) {

**try** {

sessionFactory.getCurrentSession().delete(blog);

**return** **true**;

}

**catch**(Exception ex) {

ex.printStackTrace();

**return** **false**;

}

}

}

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

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

**import** com.coll.OnlineCollaboration.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** blogCommentId) {

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

}

@Override

**public** **boolean** addBlogComments(BlogComments blogComments) {

**try** {

sessionFactory.getCurrentSession().save(blogComments);

**return** **true**;

}

**catch**(Exception ex) {

ex.printStackTrace();

**return** **false**;

}

}

@Override

**public** **boolean** updateBlogComments(BlogComments blogComments) {

**try** {

sessionFactory.getCurrentSession().update(blogComments);

**return** **true**;

}

**catch**(Exception ex) {

ex.printStackTrace();

**return** **false**;

}

}

@Override

**public** **boolean** deleteBlogComments(BlogComments blogComments) {

**try** {

sessionFactory.getCurrentSession().delete(blogComments);

**return** **true**;

}

**catch**(Exception ex) {

ex.printStackTrace();

**return** **false**;

}

}

}

7.Create the service interfaces inside com.coll.OnlineCollaboration.service package:

a.Create an interface named IUserService.java inside the above said package:

**package** com.coll.OnlineCollaboration.service;

**import** java.util.List;

**import** com.coll.OnlineCollaboration.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** activeUser(**int** userId);

**boolean** updateUserProfile(String file, Integer userId);

List<User> getAlldeactiveUser();

}

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

**package** com.coll.OnlineCollaboration.service;

**import** java.util.List;

**import** com.coll.OnlineCollaboration.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);

}

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

**package** com.coll.OnlineCollaboration.service;

**import** java.util.List;

**import** com.coll.OnlineCollaboration.model.BlogComments;

**public** **interface** IBlogCommentsService {

List<BlogComments> getAllBlogComments();

BlogComments getBlogCommentsById(**int** blogComemntId);

**boolean** addBlogComments(BlogComments blogComments);

**boolean** updateBlogComments(BlogComments blogComments);

**boolean** deleteBlogComments(BlogComments blogComments);

}

8.Create the service implementation classes inside

com.coll.OnlineCollaboration.serviceImpl package:

a.Create a class named UserServiceImpl.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.IUserDao;

**import** com.coll.OnlineCollaboration.model.User;

**import** com.coll.OnlineCollaboration.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** activeUser(**int** userId) {

**return** userDao.activeUser(userId);

}

@Override

**public** **boolean** deactiveuser(**int** userId) {

**return** userDao.deactiveUser(userId);

}

@Override

**public** **boolean** updateUserProfile(String file, Integer userId) {

**return** userDao.updateUserProfile(file, userId);

}

@Override

**public** List<User> getAlldeactiveUser() {

**return** userDao.getAlldeactiveusers();

}

}

b.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);

}

}

c.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.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** blogComemntId) {

**return** blogcommentsService.getBlogCommentsById(blogComemntId);

}

@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);

}

}

9.Create the controller classes inside com.coll.OnlineCollaboration.controller package:

a.Create a class named UserController.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.RequestMethod;

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

**import** com.coll.OnlineCollaboration.model.User;

**import** com.coll.OnlineCollaboration.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();

}

@GetMapping("deactive-user-list")

**public** List<User> alldeactiveUser() {

**return** userService.getAlldeactiveUser();

}

@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("activate-user/{userId}")

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

**return** userService.activeUser(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);

}

}

b.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:4200")

@RequestMapping(value="/api")

**public** **class** BlogController {

@Autowired

IBlogService blogService;

@PostMapping("save-blog")

**public** **boolean** saveUser(@RequestBody Blog blog) {

**return** blogService.addBlog(blog);

}

@GetMapping("blog-list")

**public** List<Blog> allBlogs() {

**return** blogService.getAllBlogs();

}

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

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

**return** blogService.deleteBlog(blog);

}

@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);

}

}

c.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:4200")

@RequestMapping(value="/api")

**public** **class** BlogCommentsController {

@Autowired

IBlogCommentsService blogcommentsService;

@PostMapping("save-BlogComments")

**public** **boolean** saveBlog(@RequestBody BlogComments blogcomments) {

**return** blogcommentsService.addBlogComments(blogcomments);

}

@GetMapping("blogcomments-list")

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

**return** blogcommentsService.getAllBlogComments();

}

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

**public** **boolean** deleteBlogComments(@PathVariable("BlogComments") BlogComments blogcommentsId) {

**return** blogcommentsService.deleteBlogComments(blogcommentsId);

}

@GetMapping("blogcomments/{blogcommentsId}")

**public** BlogComments BlogCommentsById(@PathVariable("blogcommentsId") **int** blogcommentsId) {

**return** blogcommentsService.getBlogCommentsById(blogcommentsId);

}

}

10.Save All

11.Right click on OnlineCollaborationApplication.java class and Run as Java Application.

**package** com.coll.OnlineCollaboration;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

**public** **class** OnlineCollaborationApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(OnlineCollaborationApplication.**class**, args);

}

}

**Project:Frontend:**

**Angular Project: (OnlineCollaborationAngular)**

1. Open Visual Studio Code. Create new project inside Angular\_Workspace folder. Set the name of the project 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 { NgModule } from '@angular/core';

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

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

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

import {DataTablesModule} from 'angular-datatables';

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

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

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

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

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

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

@NgModule({

  declarations: [

    AppComponent,

    RegisterUserComponent,

    LoginUserComponent,

    AddBlogComponent,

    ApproveBlogComponent,

    ViewBlogComponent,

    ActivateUserComponent,

    UserListComponent,

    ActivateUserComponent

  ],

  imports: [

    BrowserModule,

    AppRoutingModule,

    FormsModule,

    DataTablesModule,

    HttpClientModule,

    ReactiveFormsModule,

  ],

  providers: [UserService,BlogService],

  bootstrap: [AppComponent]

})

export class AppModule { }

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

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

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

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

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

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

import { UserListComponent } from './components/user-list/user-list.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-primary" role="button">View Users</a>

      </li>

      <li class="nav-item">

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

      </li>

      <li class="nav-item">

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

    </ul>

  </nav>

</div>

<router-outlet></router-outlet>

1. 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;

}

1. 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'});

  }

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

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

  }

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

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

}

}

1. 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();

  }

}

1. 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="password" 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="password" 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>

1. Open user-list.component.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');

  }

  changeisUpdate(){

    this.isupdated=false;

  }

}

1. Open login-user.component.ts and modify as following:

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

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

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

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

@Component({

  selector: 'app-login-user',

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

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

})

export class LoginUserComponent implements OnInit {

  user:User=new User();

  submitted=false;

  constructor(private userservice:UserService) { }

  ngOnInit(): void {

    this.submitted=false;

  }

  loginform=new FormControl({

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

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

  });

  login(login){

    this.user=new User();

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

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

    }

    save(){

      this.userservice.CreateUser(this.user)

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

      this.user=new User();

    }

    get Username(){

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

    }

    get Password(){

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

    }

      loginForm(){

        this.submitted=false;

        this.loginform.reset();

      }

  }

1. Open login-user.component.html and modify as following:

<h3>Login Here</h3>

<div class="row">

    <div class="col-sm-4"></div>

    <div class="col-sm-4">

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

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

                <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="password" 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;">

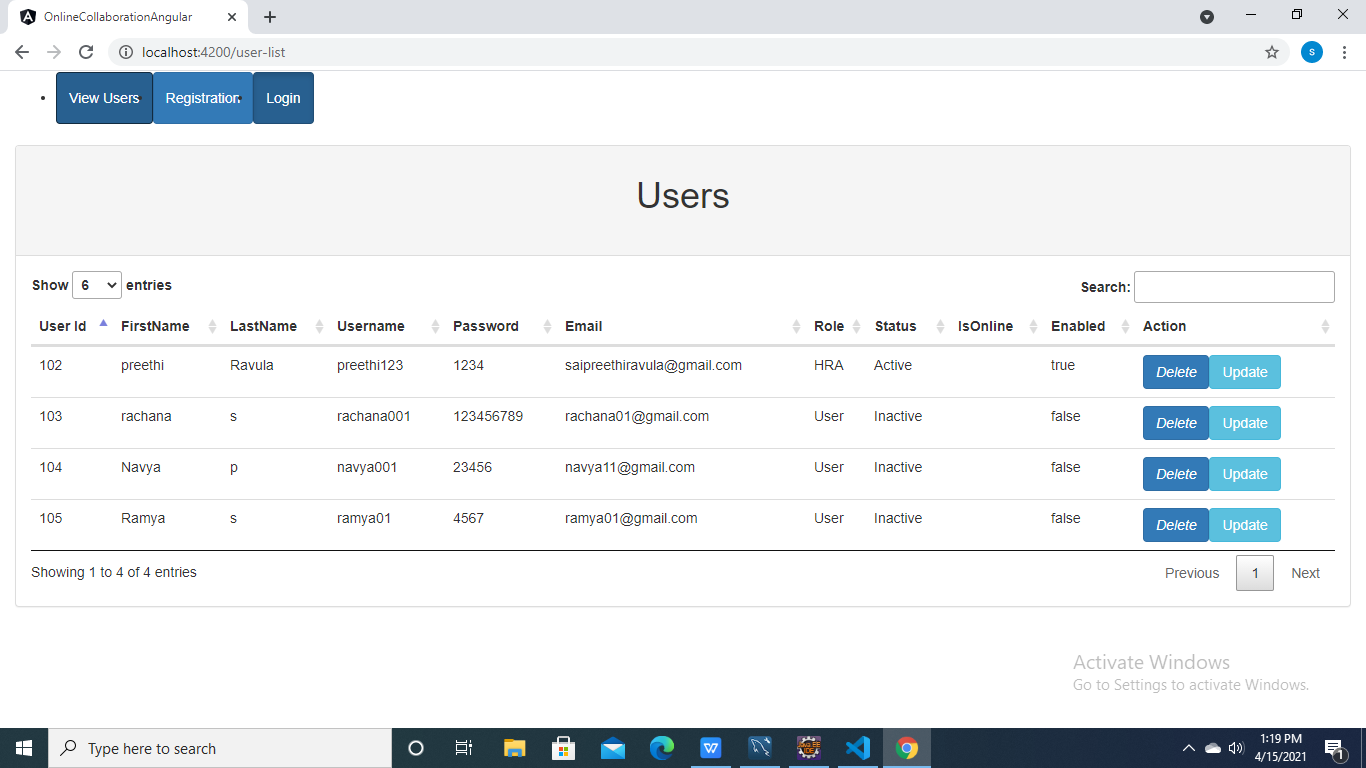
                        <span \*ngIf="Password.error.required">Password is required</span>

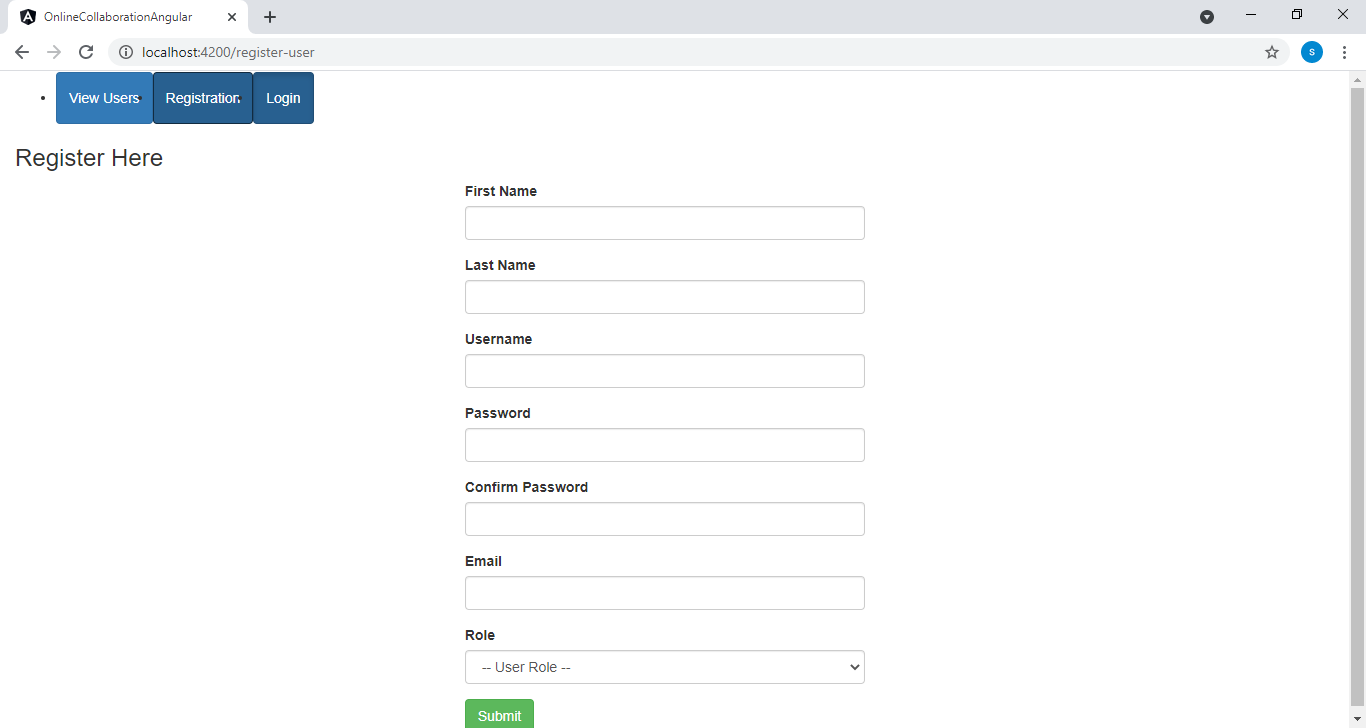
                    </div>

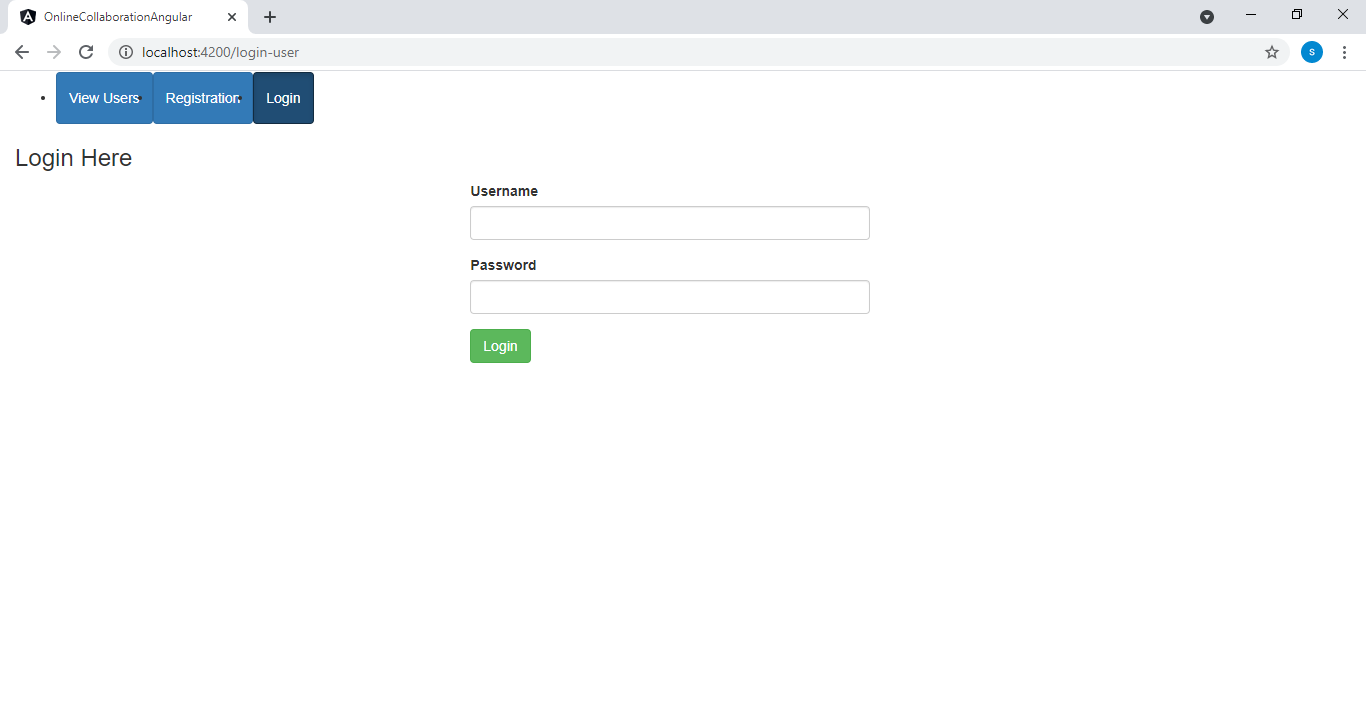
                </div>

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

**Output:**







**REFERENCES**

* NIIT Project Guide
* Google