

Project Report

on

Chat Management System

Developed By:

Name: Anzla Sharma Reg. No.: **R201171400503**

**Batch Code: S210050**

**Start Date: 10/08/2020**

**End Date: 31/08/2020**

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

**Names of the Developer: Anzla Sharma**

**Date of Submission: 31/08/2020**

**CERTIFICATION**

**This is to certify that this report, titled Chat Management System embodies the original work done by Anzla Sharma, in partial fulfillment of his course requirement at NIIT.**

**Coordinator: Ms. Lopamudra Bera**

**ACKNOWLEDGEMENT**

My efforts bore fruit with the completion of this project. However, there are many others who share the reward of this effort simply because it would never have been this good without their help. I acknowledge the cooperation, encouragement and austerity of Ms. Lopamudra Bera whose guidance did half the magic of keeping me thrilled throughout this project.

ABSTRACT

This project manages the entire process of allocating the project to the user as per his/ her working skill and to communication further to his/her teammates.

This application provides a single window system to all users and to admin of a company to cater the skill specific requirements emerging in Projects.

CONFIGURATION

Hardware: X64 Based PC

Software:

* Windows 10
* RAM: 128 MB
* Disk space: 124 MB for JRE; 2 MB for Java Update

Contents

|  |  |
| --- | --- |
| **Sr. No.** | **Description** |
| **1.** | Introduction  -Objective |
| **2.** | Software Requirement Specifications |
| **3.** | Project Requirement Analysis  -Project Plan  -System Architecture |
| **4.** | Project Design  -High Level Design  -Low Level Design  -User Interface Design  -System Input and Output Design  -Data Model |
| **5.** | Project Code |
| **6.** | Benefits |

**OBJECTIVE**

This project manages the entire process of allocating the project to the user as per his/ her working skill and to communication further to his/her teammates.

This application provides a single window system to all users and to admin of a company to cater the skill specific requirements emerging in Projects.

**System Requirement Specification (SRS)**

|  |  |
| --- | --- |
| **S.No.** | **System Requirement** |
| 1 | Login screen for all users and admin. |
| 2 | Application should have an option for admin to activate newly registered user. |
| 3 | Application should have an option for admin to make a workspace for users . |
| 4 | Application should have an option for admin to request users to be a part of workspace. |
| 5 | Application should have an option for user to decline the request generated by admin. |
| 6 | Application should have an option for user to view his own |
| 7 | There should an provision for user to raise request to admin for changing the base information |
| 8 | Application should have an option for admin to communicate the user directly. |
| 9 | Application should have an option for user to have onwe to one communication with other users. |
| 10 | Application should have an option for admin to see all users detail. |
| 11 | In this application admin should have a right to decline a user. |
| 12 | There should be login /logout facility for user and admin. |

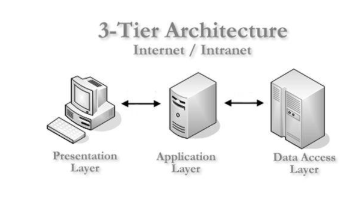
**Project Plan**

Phase wise plan -

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Project Phases | W1 | W2 | W3 | W4 | W5 | W6 | W7 |
| Requirement gathering |  |  |  |  |  |  |  |
| High Level Design |  |  |  |  |  |  |  |
| Low Level Design |  |  |  |  |  |  |  |
| UI design |  |  |  |  |  |  |  |
| Code Development |  |  |  |  |  |  |  |
| testing |  |  |  |  |  |  |  |
| Roll Out |  |  |  |  |  |  |  |

This project was planned to be completed in 7 Weeks’ time.

**System Architecture**



An **N-Tier Application** program is one that is distributed among three or more separate computers in a distributed network.

The most common form of n-tier is the 3-tier Application, and it is classified into three categories.

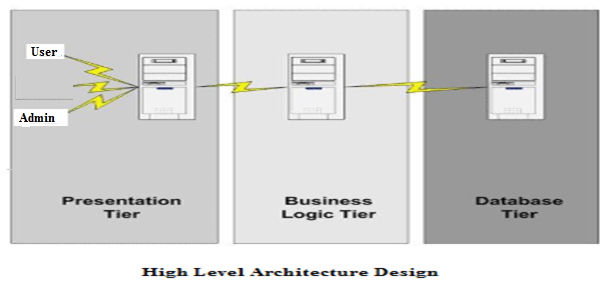
* User interface programming in the user's computer
* Business logic in a more centralized computer, and
* Required data in a computer that manages a database.

This architecture model provides Software Developers to create Reusable application/systems with maximum flexibility.

**3-tier architecture**has three different layers.

* Presentation layer
* Business Logic layer
* Database layer

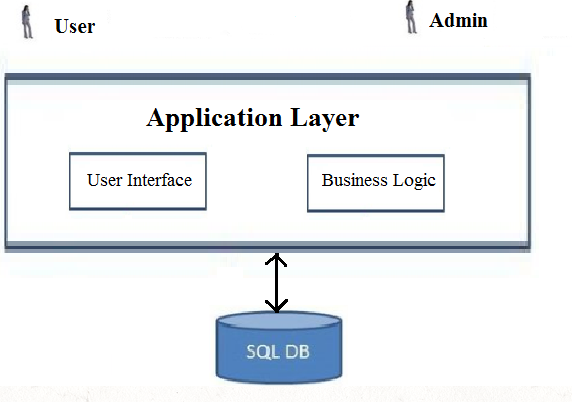
**High Level Architecture Design**

****

In the above figure the users namely EMP, HR , PM interact with the presentation tier of the application. After that the control is passed to business logic tier which interacts with database to fetch the required data. Once the required data is fetched the user can see the result on the presentation tier.

**High Level Design**

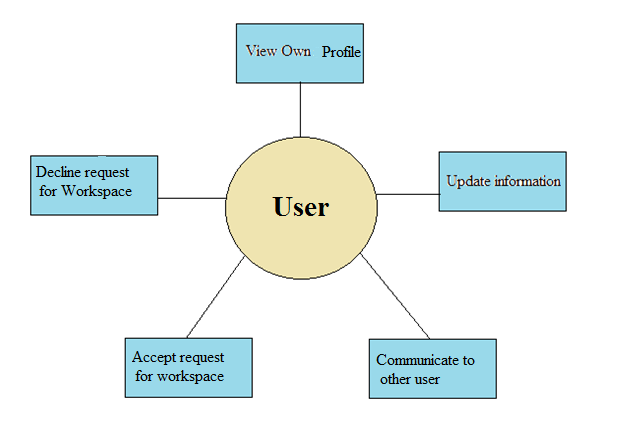
High-level design explains the architecture that would be used for developing a software product. The architecture diagram provides an overview of an entire system, identifying the main components that would be developed for the product and their interfaces



The user/s interact with the application layer via UI, which passes the request to the Business logic layer

The business logic layer interacts with the database tier where all records are stored to be fetched.

. **Low Level Design**

Here is the low level design of User. In this application the user can perform following task:

* User can view his/her own information.
* User can communicate with other users.
* User can accept the request for workspace generated by the admin.
* User can decline the request for workspace generated by the admin.
* Employees can logout his/her account for the safety purpose.

User can accept the request for workspace generated by the admin

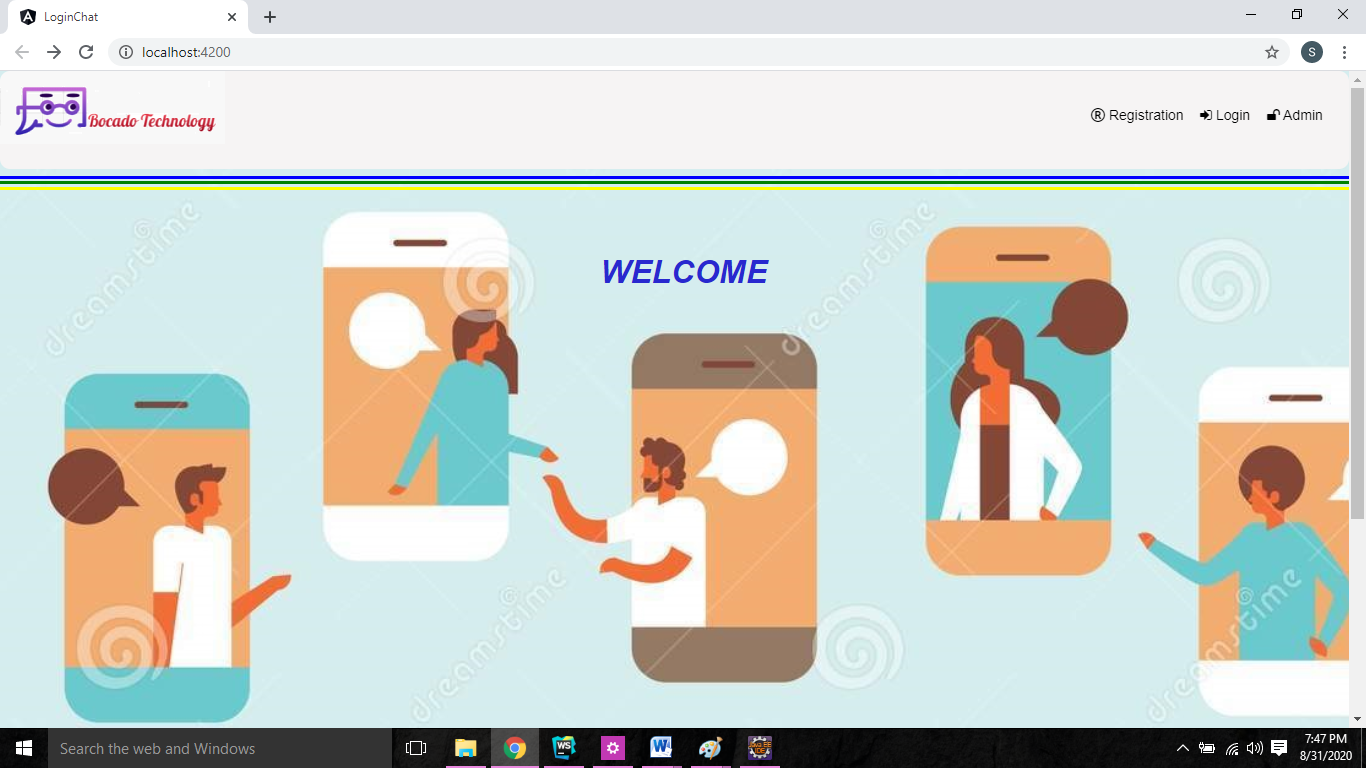
Here is the low level design of the admin. In this application the admin can perform following task:

* HR can view employee information.
* Admin can generate a request for user so he /she can be part of a group(Workspace).
* Admin allows new user to be the part of the application.
* Admin view users information.
* Admin can communication directly to users by one-one or in a group.
* HR can logout his/her account for the safety purpose.

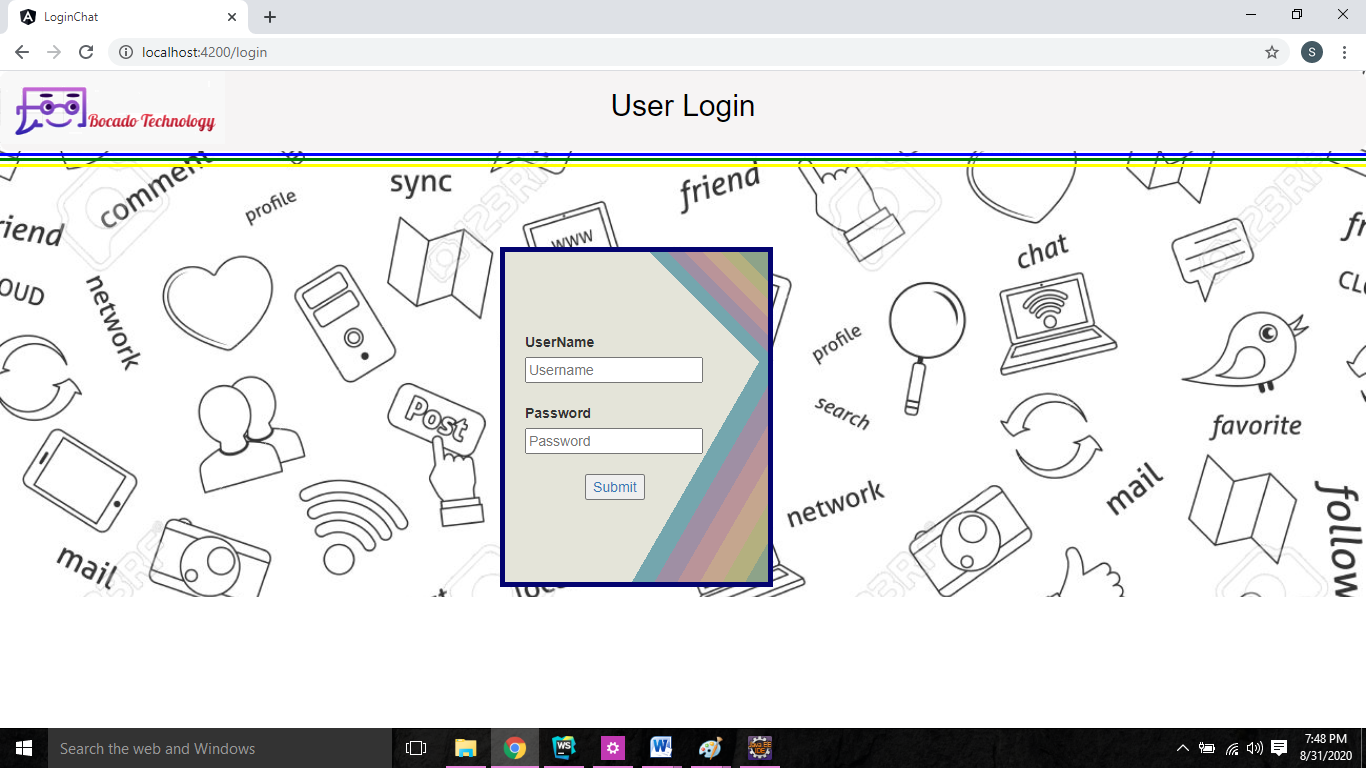
**User Interface Design**

The login page of users and admin are not common. The user ID and password are stored in database while the user registered him/ her at the first time.

Main Page

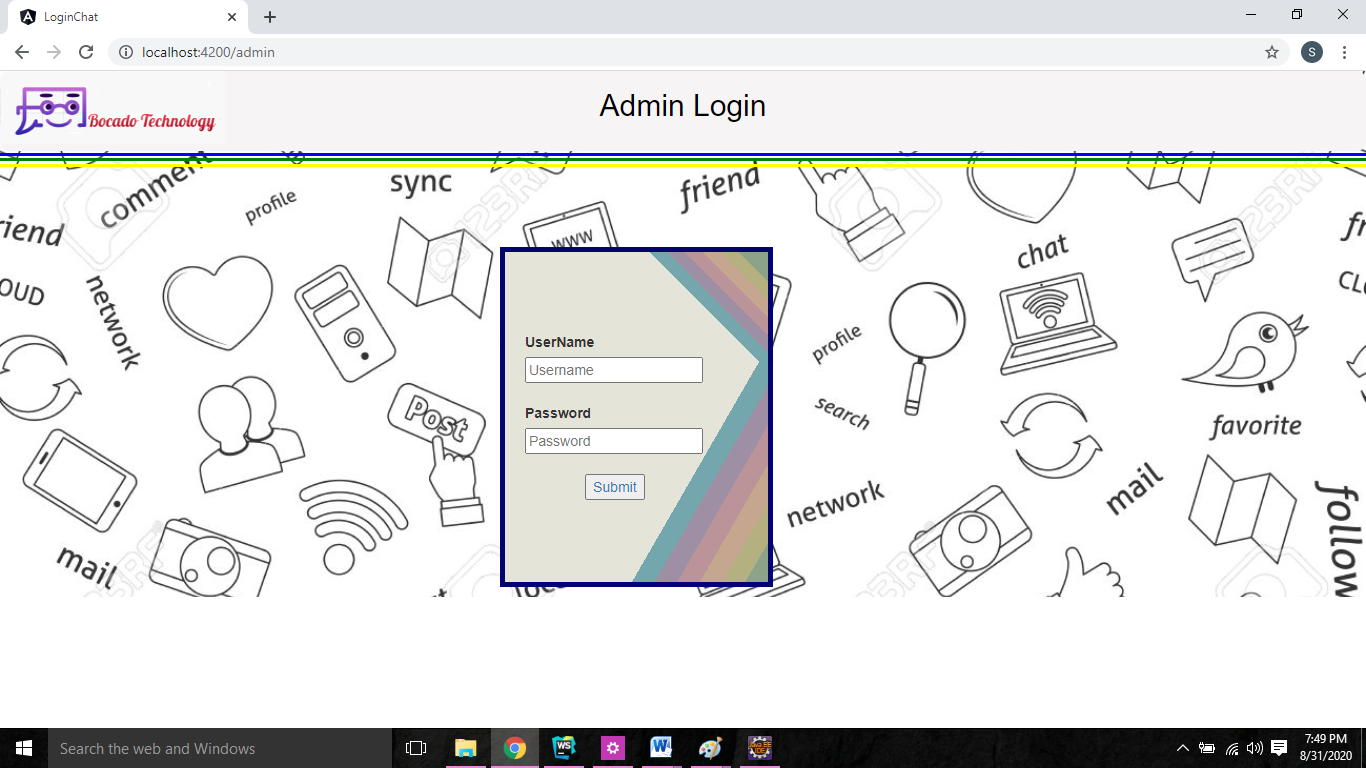


User Login



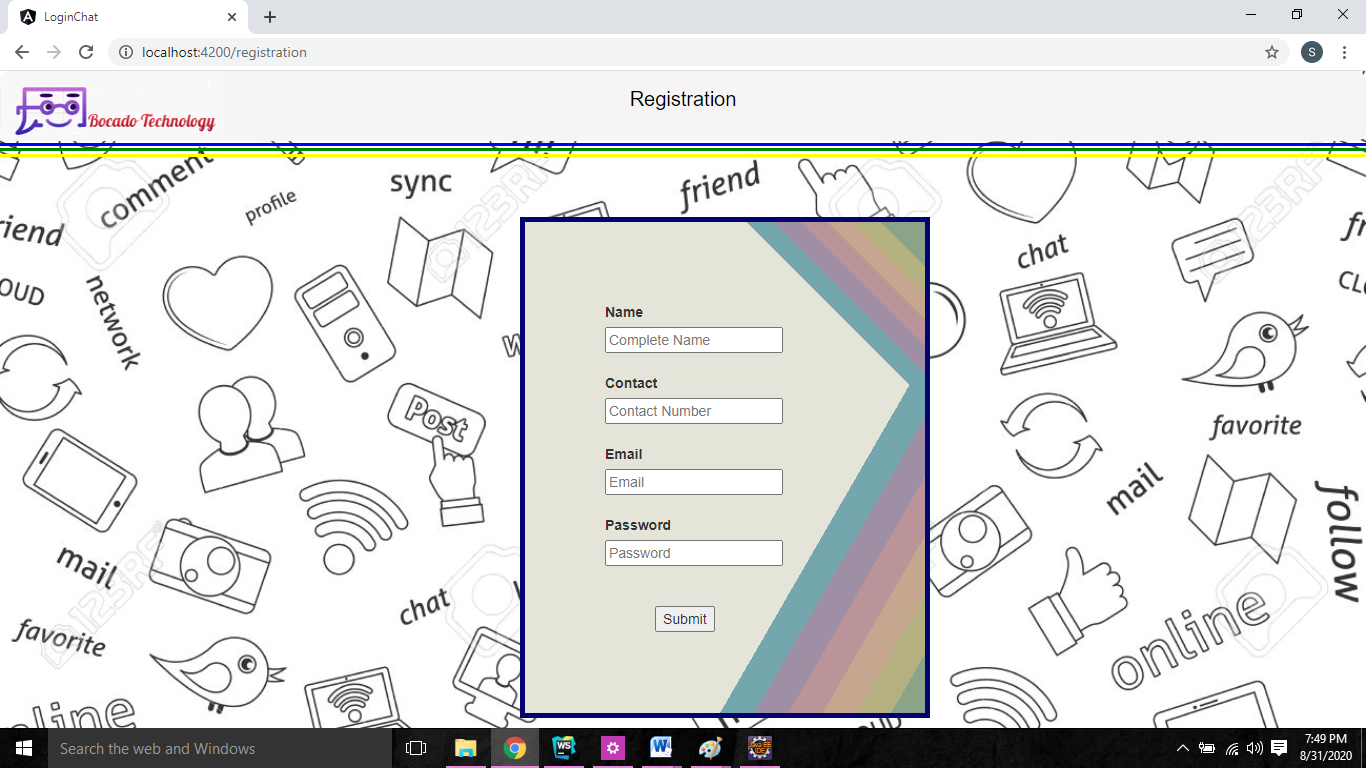
With the credentials user can login to his/her account.

Admin Login



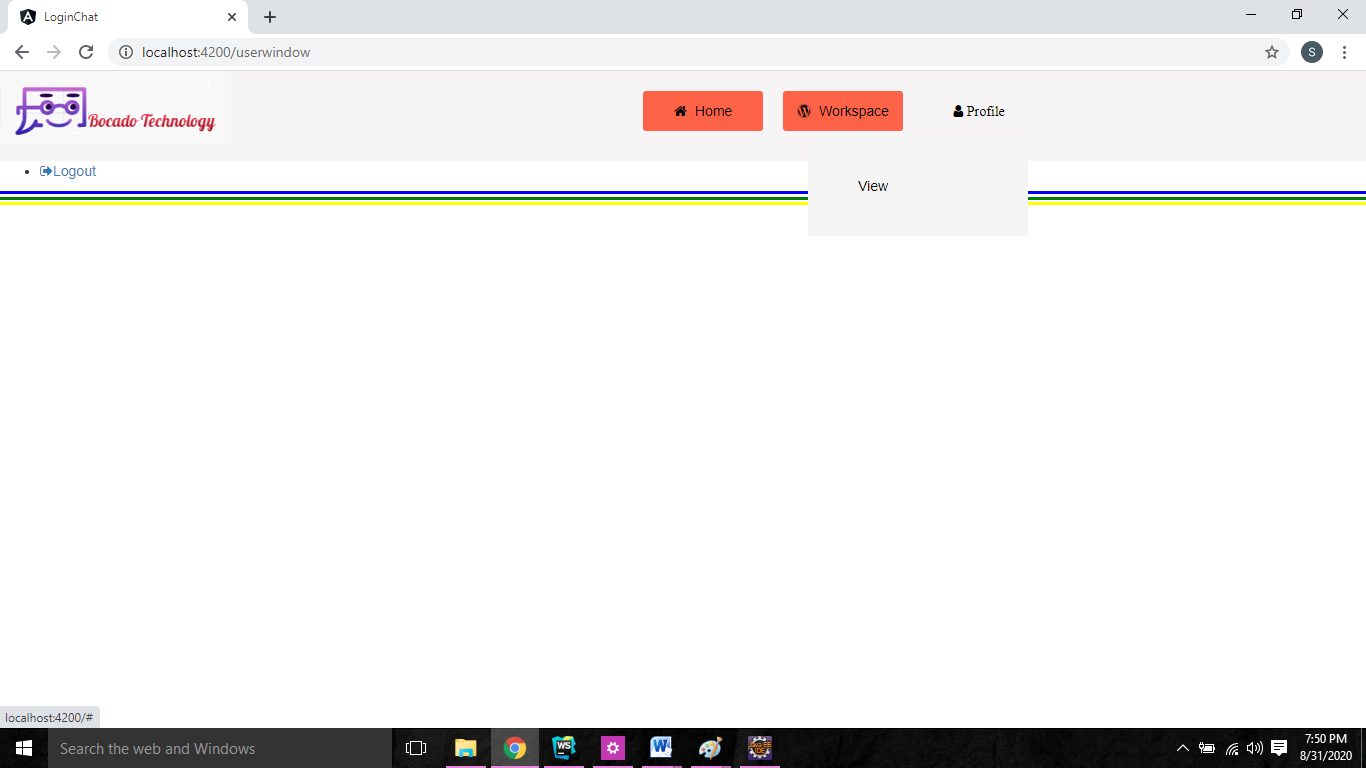
Admin can login using the credentials.

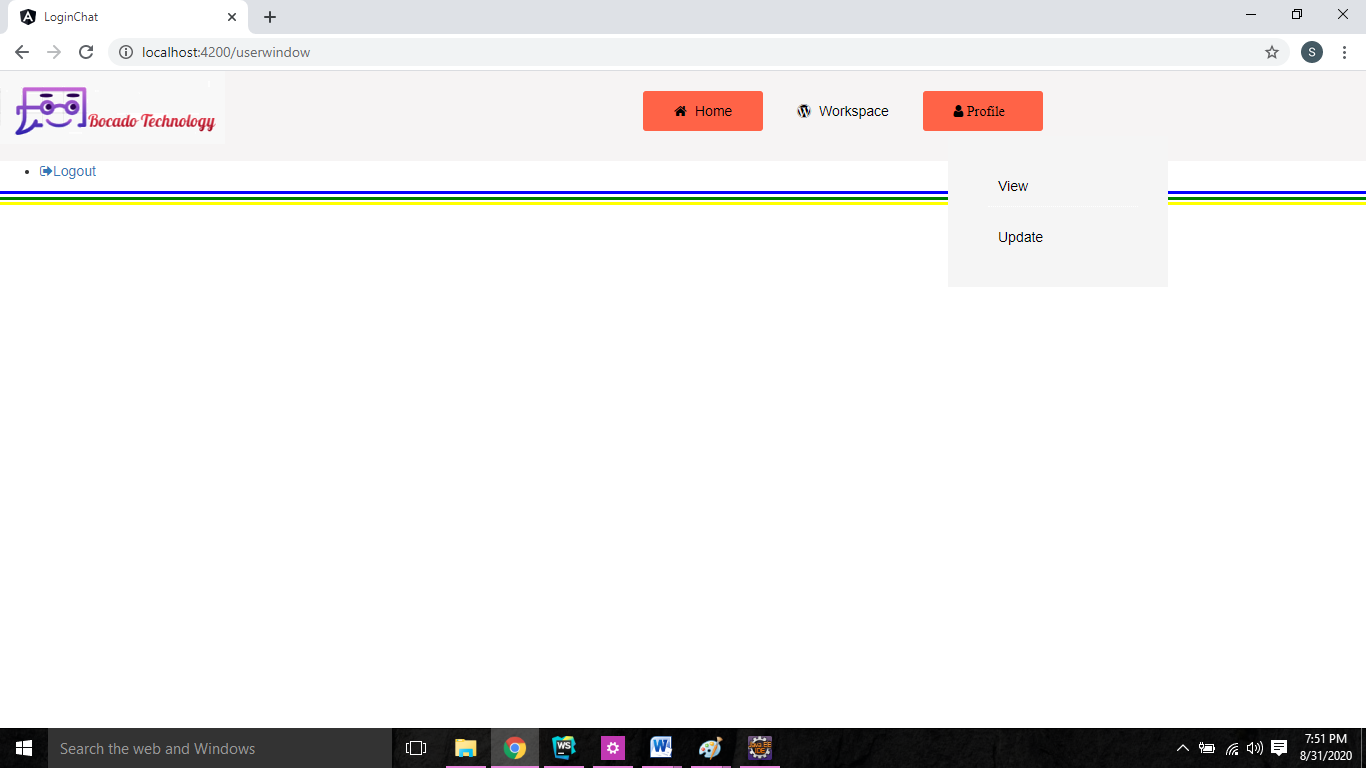
Registration Page



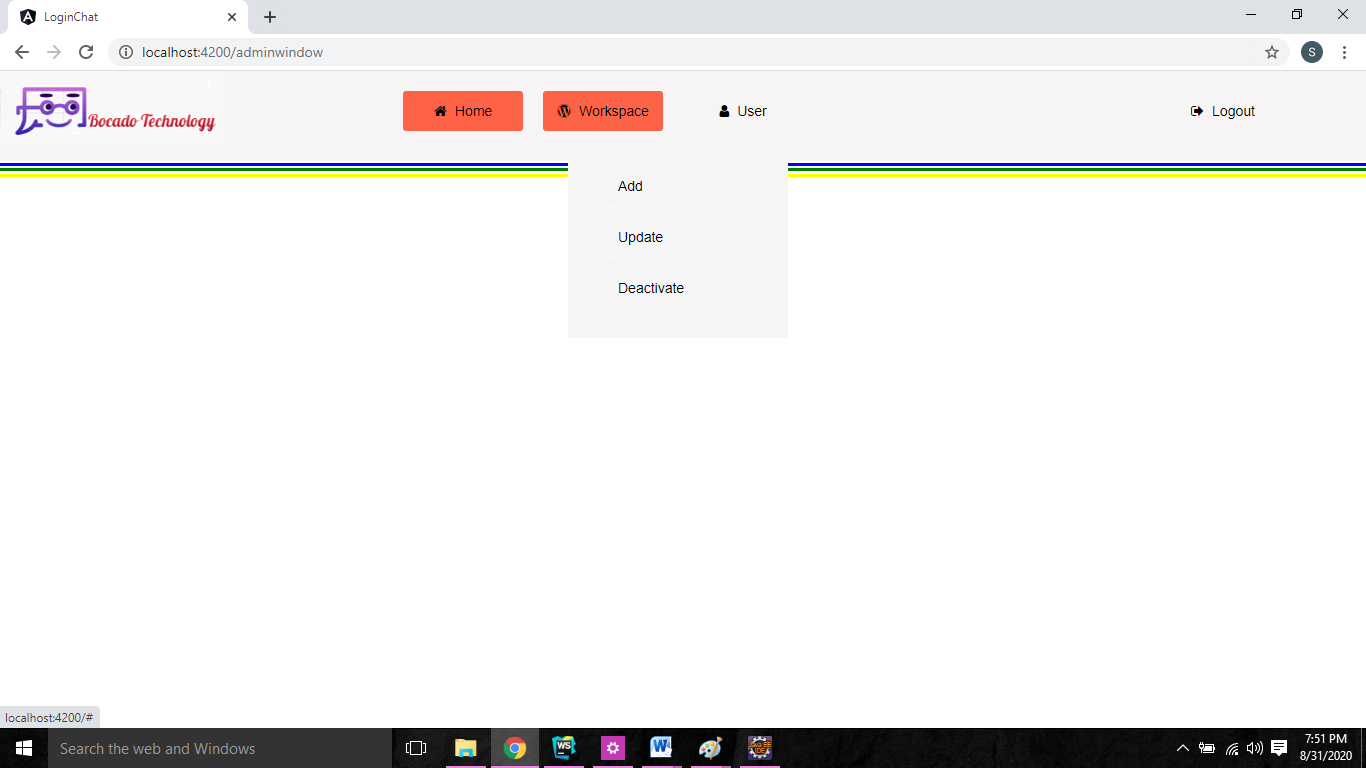
New users can register here the required information.

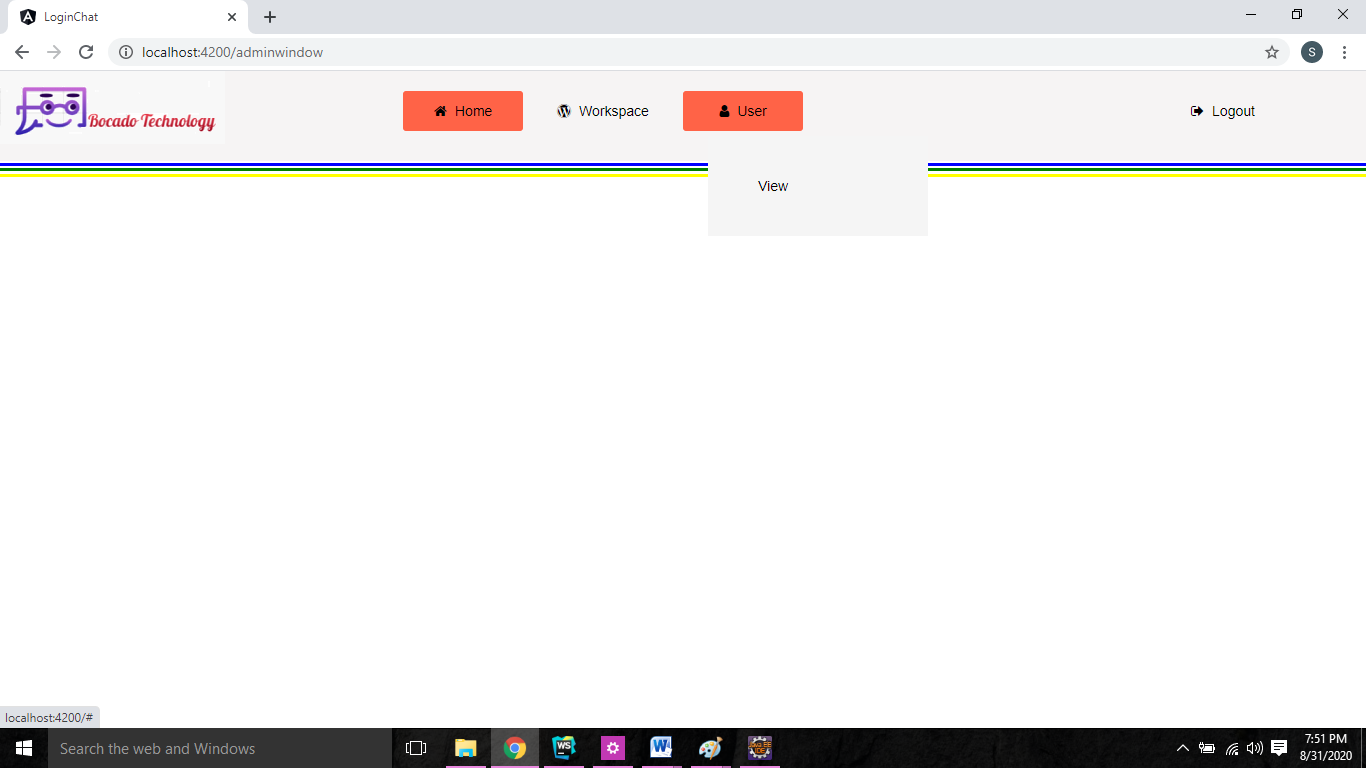
Login Window



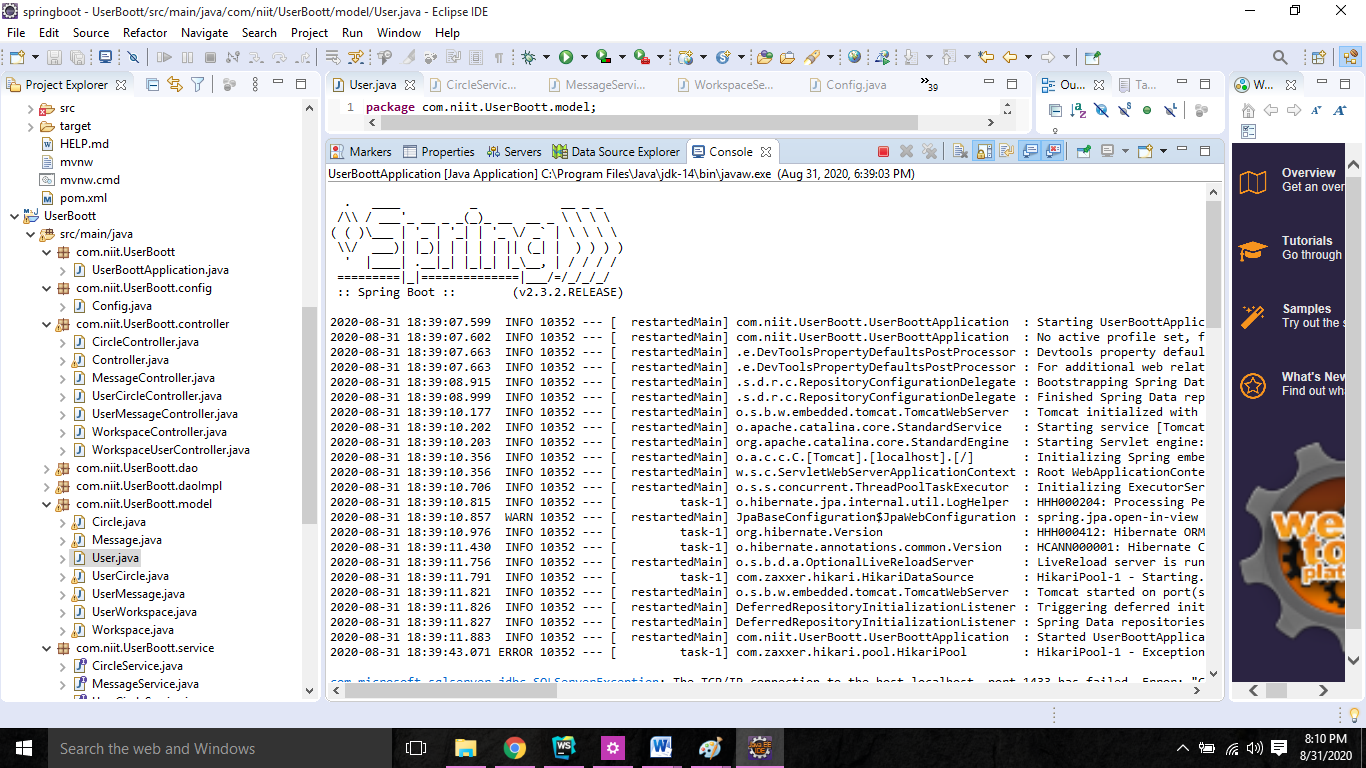


Admin Window





Eclipse output



Sample Project Code –

**Configuration code**

package com.niit.UserBoott.config;

import java.util.Properties;

import javax.sql.DataSource;

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

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.transaction.annotation.EnableTransactionManagement;

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

@Configuration

@EnableTransactionManagement

@EnableAutoConfiguration(exclude = { HibernateJpaAutoConfiguration.class})

@ComponentScans(value = { @ComponentScan("com.niit.UserBoott"),

@ComponentScan("Model"),

@ComponentScan("Controller"),

@ComponentScan("DAO"),

@ComponentScan("Miscallaneous"),

@ComponentScan("Service")})

public class Config {

@Value("${db.driver}")

private String DB\_DRIVER;

@Value("${db.password}")

private String DB\_PASSWORD;

@Value("${db.url}")

private String DB\_URL;

@Value("${db.username}")

private String DB\_USERNAME;

@Value("${hibernate.dialect}")

private String HIBERNATE\_DIALECT;

@Value("${hibernate.show\_sql}")

private String HIBERNATE\_SHOW\_SQL;

@Value("${hibernate.hbm2ddl.auto}")

private String HIBERNATE\_HBM2DDL\_AUTO;

@Value("${entitymanager.packagesToScan}")

private String ENTITYMANAGER\_PACKAGES\_TO\_SCAN;

@Bean

public LocalSessionFactoryBean sessionFactory() {

LocalSessionFactoryBean sessionFactory = new LocalSessionFactoryBean();

sessionFactory.setDataSource(dataSource());

sessionFactory.setPackagesToScan(ENTITYMANAGER\_PACKAGES\_TO\_SCAN);

Properties hibernateProperties = new Properties();

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

hibernateProperties.put("hibernate.show\_sql", HIBERNATE\_SHOW\_SQL);

hibernateProperties.put("hibernate.hbm2ddl.auto", HIBERNATE\_HBM2DDL\_AUTO);

sessionFactory.setHibernateProperties(hibernateProperties);

return sessionFactory;

}

@Bean

public DataSource dataSource() {

DriverManagerDataSource dataSource = new DriverManagerDataSource();

dataSource.setDriverClassName(DB\_DRIVER);

dataSource.setUrl(DB\_URL);

dataSource.setUsername(DB\_USERNAME);

dataSource.setPassword(DB\_PASSWORD);

return dataSource;

}

@Bean

public HibernateTransactionManager transactionManager() {

HibernateTransactionManager txManager = new HibernateTransactionManager();

txManager.setSessionFactory(sessionFactory().getObject());

return txManager;

}

@Bean

public InternalResourceViewResolver jspViewResolver() {

InternalResourceViewResolver resolver= new InternalResourceViewResolver();

resolver.setPrefix("/views/");

resolver.setSuffix(".jsp");

return resolver;

}

}

**Model Class**

package com.niit.UserBoott.model;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table(name="Users")

public class User {

@Id

@GeneratedValue(strategy=GenerationType.IDENTITY)

private int User\_id;

private String User\_name;

private String User\_email;

private String User\_password;

private String User\_Phone;

public int getUser\_id() {

return User\_id;

}

public void setUser\_id(int user\_id) {

User\_id = user\_id;

}

public String getUser\_name() {

return User\_name;

}

public void setUser\_name(String user\_name) {

User\_name = user\_name;

}

public String getUser\_email() {

return User\_email;

}

public void setUser\_email(String user\_email) {

User\_email = user\_email;

}

public String getUser\_password() {

return User\_password;

}

public void setUser\_password(String user\_password) {

User\_password = user\_password;

}

public String getUser\_Phone() {

return User\_Phone;

}

public void setUser\_Phone(String user\_Phone) {

User\_Phone = user\_Phone;

}

@Override

public String toString() {

return "User [User\_id=" + User\_id + ", User\_name=" + User\_name + ", User\_email=" + User\_email

+ ", User\_password=" + User\_password + ", User\_Phone=" + User\_Phone + "]";

}

}

Benefits of Implementing Chat Management System(CMS)

* Real time text preview.
* Improves customer service and loyalty.
* Team Management easier.
* Transfer of data is easier.

---------------- End Of Document ------------------