Project: Login and Registration Application

Developed by: Karthiga Gujuluva Ravichandran
Email: (karthiga.ravic@hcl.com)
Created on Date: 01/31/2021

Prototype:

The prototype of Login and Registration Application consists of a welcome screen, register and login screen with functionalities in each screen. The user interaction will be through a webpage of link (http://localhost:8080/user/) which leads to either registration or login screens and the final Application will be developed in various sprints based this prototype. The prototype explains the functionalities and format of the final project.

Sprint Plan:

Login and Registration Application project is divided into **3 sprints**. Each sprint consists of **1 week** of time. The tasks achieved in the sprints are given below:

Sprint 1:

Time allocated for this sprint is **40 hours**. The tasks involved in this sprint are:

Tasks:

1. Development of Welcome screen and Login screen.
2. Welcome screen should have 2 options i.e., Registration button and login button.
3. Login screen has 2 inputs namely username and password and a submit button to proceed to login success page.
4. The username and password should be required fields and can accept all characters.
5. Login button searches if user already exists in the database and proceeds to login success screen.
6. Database is created for the application and fed with initial data.
7. Linking all screens and the database through HibernateUtil.java file.
8. Servlet is used to create welcome and login screens.
9. Dao, model and servlets are used to get data from DB, validate and display on the webpage

Sprint 2:
Time allocated for this sprint is **40 hours**. The tasks involved in this sprint are:

Tasks:
1. Development of registration screen that has first name, last name, username, password, contact number, email and a register button.
2. Register button leads to login screen.
3. All are required fields.
4. Contact number accepts only 12 numbers.
5. Error messages are thrown in login screen if incorrect number is given in contact number.
6. Save method is created to save the data to table.
7. Find user functionality checks if user already exists in table.
8. Appropriate messages are thrown to the user if save happens or fails.

Sprint 3:
Time allocated for this sprint is **40 hours**. The tasks involved in this sprint are:

Tasks:
1. Login screen enhancement like throwing of error messages is done in this phase.
2. Incorrect password, User duplication messages, incorrect username and leading to registration screen and handling of unexpected errors with custom exceptions should be done in this phase.
3. Login success page is created and the contents of the page are fixed.
4. Fixing the bugs from previous sprints.
5. Development of enhancements like developing of better UI, alignment, preventing code redundancy, increasing performance using streams, lambda and regex expressions.
6. Logging should be added to the project for future usage.
7. Redundancy of code should be checked and commenting on the code for better usage
8. Creation of documentation and JUnit tests for the project.

Login and Registration Application - Link

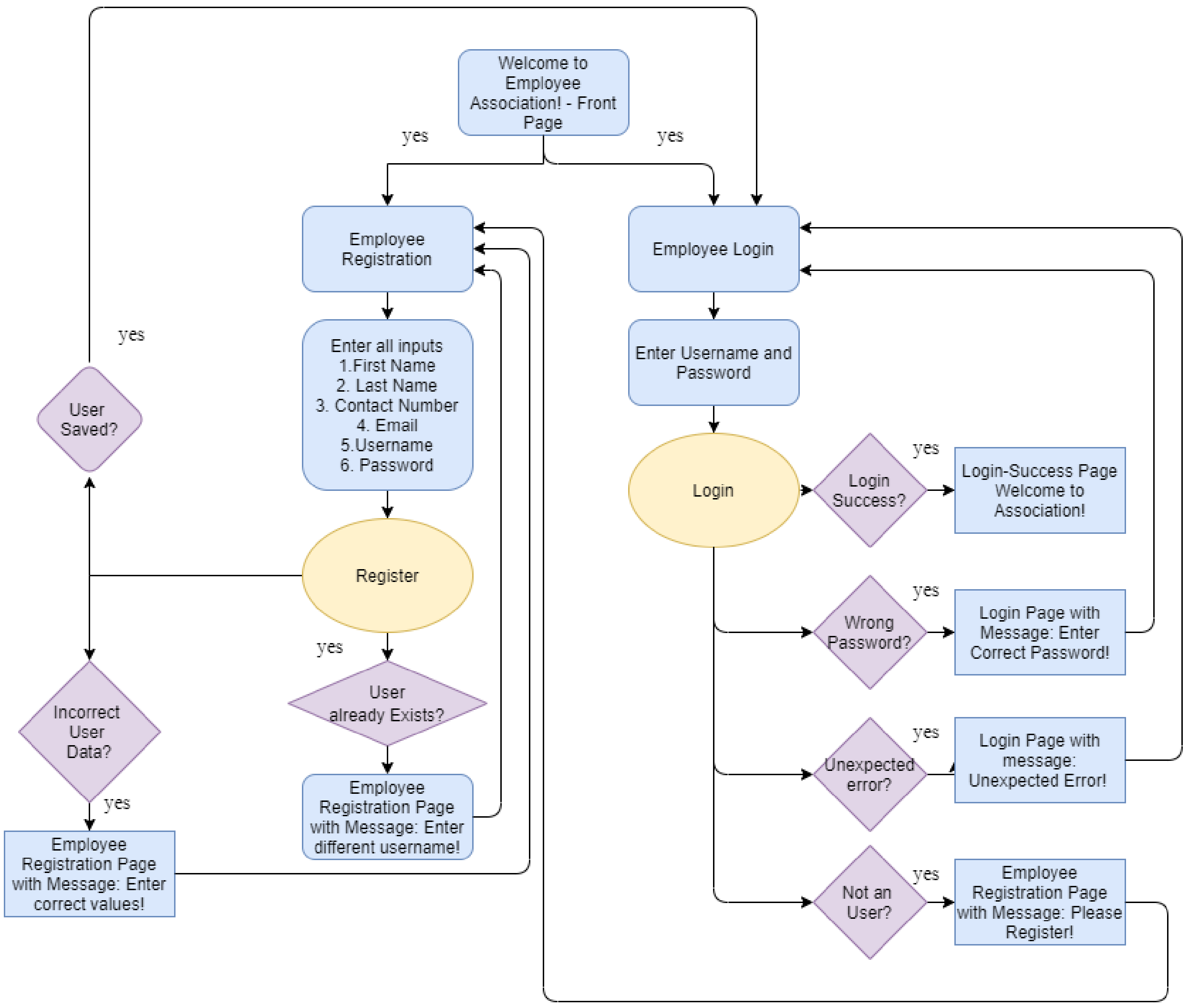
https://github.com/Karthiga-web/userLoginRegisterPhaseTwo.git is the repository link for my Login and Registration Application in GitHub.

Algorithm:

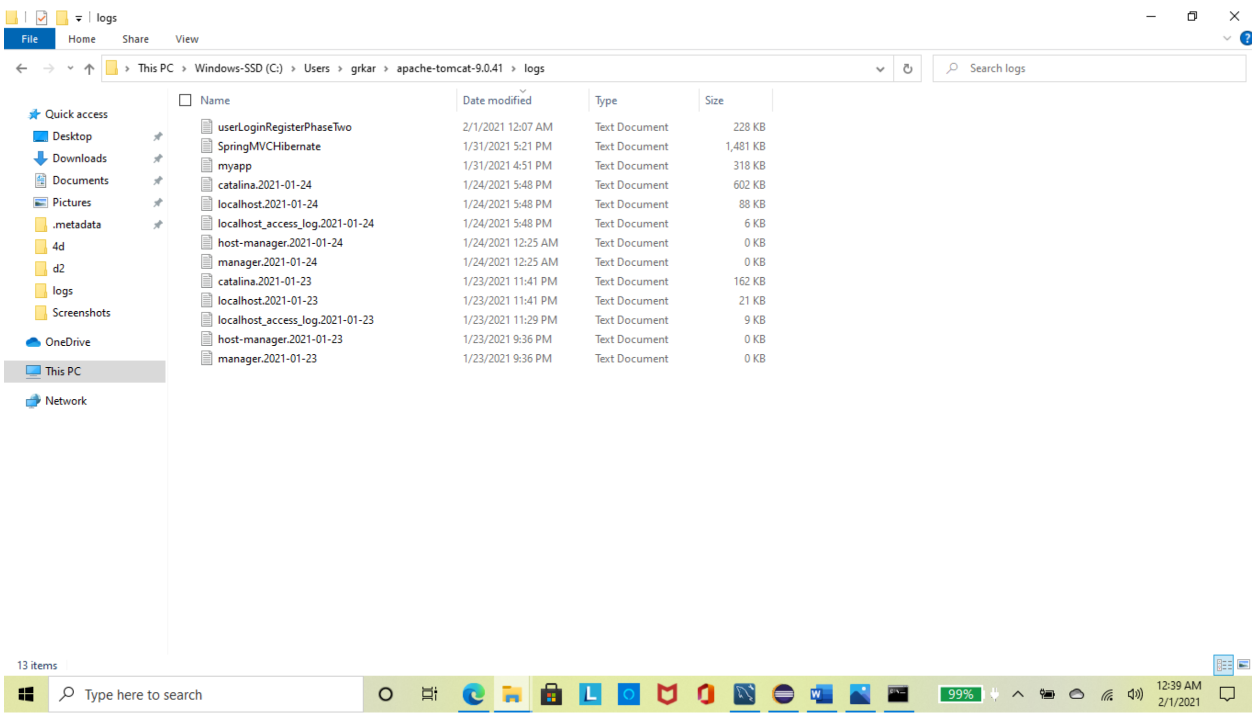
1. The project opens with Welcome Screen that has Register and Login buttons
2. When user clicks Login button, the page leads to login screen
3. Login screen has username and password inputs and submit button
4. When user gives correct username and password, it takes to successful login page
5. When user gives incorrect username and correct password, it takes to successful registration page that displays to register as being a new user.
6. When user gives correct username and incorrect password, it takes to login page with error that wrong password is given
7. For any unexpected error caused by the system, the login page appears with error message to try again
8. When user clicks register button in Welcome screen, it takes to registration page
9. The registration page asks the user to give first name, last name, contact number, email, password and asks the user to submit
10. If user enter all correct data, then the data is successfully saved in the DB and leads to login page
11. If user enter incorrect contact number like other than numbers, then registration page displays an error to enter correct values in the fields
12. The successful login page appears at last and the application is completed
13. Logger is incorporated in this project for future usage.

Flowchart:

Login and Registration Application


Welcome to Employee Association! - Front Page
yes
yes
Employee Registration
Employee Login
Enter all inputs
1.First Name
2. Last Name
3. Contact Number
4. Email
5.Username
6. Password
Enter Username and Password
yes
User Saved?
Login
Login Success?
yes
Login-Success Page Welcome to Association!
Register
Wrong Password?
yes
Login Page with Message: Enter Correct Password!
yes
Incorrect User Data?
User already Exists?
Unexpected error?
yes
Login Page with message: Unexpected Error!
yes
Employee Registration Page with Message: Enter different username!
Employee Registration Page with Message: Enter correct values!
Not an User?
yes
Employee Registration Page with Message: Please Register!

Logger Saving Folder:



logs

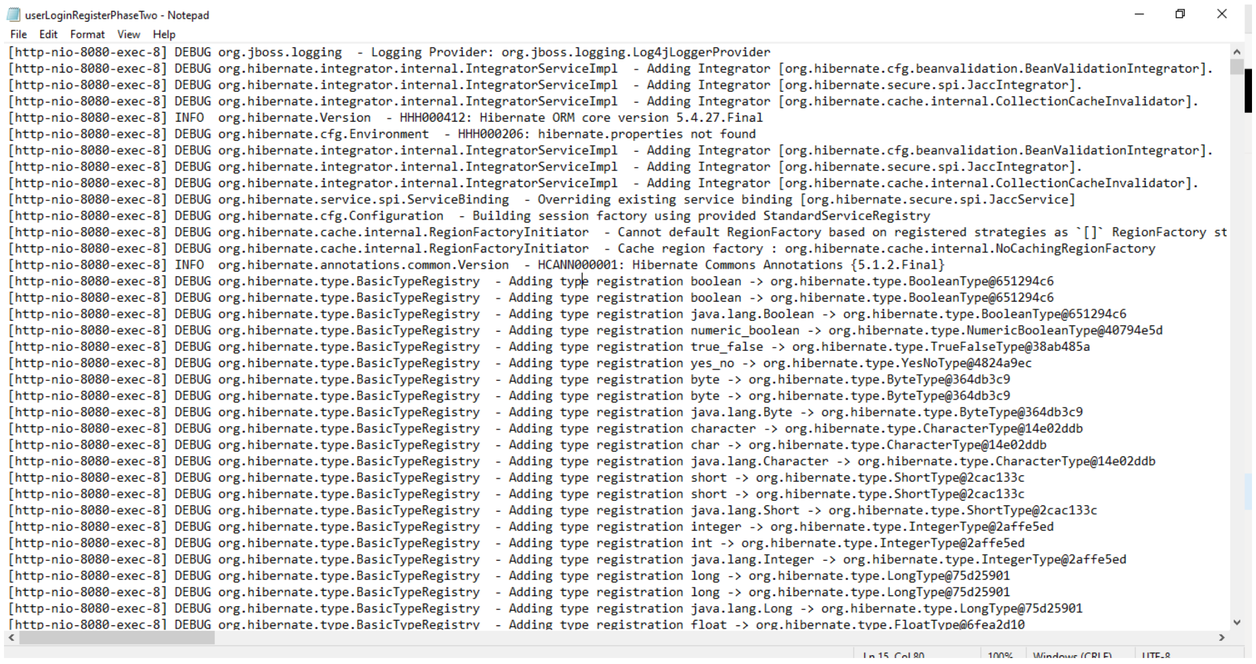
File Home Share View

This PC › Windows-SSD (C:) › Users › grkar › apache-tomcat-9.0.41 › logs

Name	Date modified	Type	Size
userLoginRegisterPhaseTwo	2/1/2021 12:07 AM	Text Document	228 KB
SpringMVCHibernate	1/31/2021 5:21 PM	Text Document	1,481 KB
myapp	1/31/2021 4:51 PM	Text Document	318 KB
catalina.2021-01-24	1/24/2021 5:48 PM	Text Document	602 KB
localhost.2021-01-24	1/24/2021 5:48 PM	Text Document	88 KB
localhost_access_log.2021-01-24	1/24/2021 5:48 PM	Text Document	6 KB
host-manager.2021-01-24	1/24/2021 12:25 AM	Text Document	0 KB
manager.2021-01-24	1/24/2021 12:25 AM	Text Document	0 KB
catalina.2021-01-23	1/23/2021 11:41 PM	Text Document	162 KB
localhost.2021-01-23	1/23/2021 11:41 PM	Text Document	21 KB
localhost_access_log.2021-01-23	1/23/2021 11:29 PM	Text Document	9 KB
host-manager.2021-01-23	1/23/2021 9:36 PM	Text Document	0 KB
manager.2021-01-23	1/23/2021 9:36 PM	Text Document	0 KB

13 items

Logger File

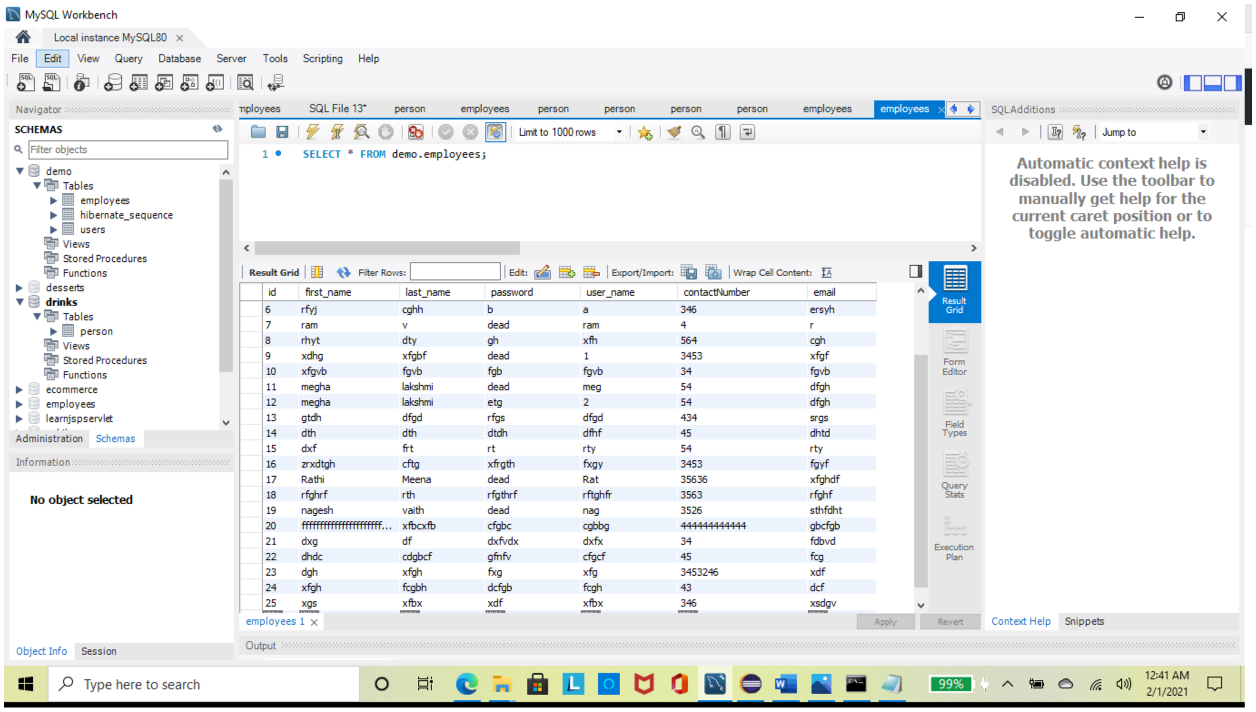


userLoginRegisterPhaseTwo - Notepad

File Edit Format View Help

```
[http-nio-8080-exec-8] DEBUG org.jboss.logging  - Logging Provider: org.jboss.logging.Log4jLoggerProvider
[http-nio-8080-exec-8] DEBUG org.hibernate.integrator.internal.IntegratorServiceImpl  - Adding Integrator [org.hibernate.cfg.beanvalidation.BeanValidationIntegrator].
[http-nio-8080-exec-8] DEBUG org.hibernate.integrator.internal.IntegratorServiceImpl  - Adding Integrator [org.hibernate.secure.spi.JaccIntegrator].
[http-nio-8080-exec-8] DEBUG org.hibernate.integrator.internal.IntegratorServiceImpl  - Adding Integrator [org.hibernate.cache.internal.CollectionCacheInvalidator].
[http-nio-8080-exec-8] INFO  org.hibernate.Version  - HHH000412: Hibernate ORM core version 5.4.27.Final
[http-nio-8080-exec-8] DEBUG org.hibernate.cfg.Environment  - HHH000206: hibernate.properties not found
[http-nio-8080-exec-8] DEBUG org.hibernate.integrator.internal.IntegratorServiceImpl  - Adding Integrator [org.hibernate.cfg.beanvalidation.BeanValidationIntegrator].
[http-nio-8080-exec-8] DEBUG org.hibernate.integrator.internal.IntegratorServiceImpl  - Adding Integrator [org.hibernate.secure.spi.JaccIntegrator].
[http-nio-8080-exec-8] DEBUG org.hibernate.integrator.internal.IntegratorServiceImpl  - Adding Integrator [org.hibernate.cache.internal.CollectionCacheInvalidator].
[http-nio-8080-exec-8] DEBUG org.hibernate.service.spi.ServiceBinding  - Overriding existing service binding [org.hibernate.secure.spi.JaccService]
[http-nio-8080-exec-8] DEBUG org.hibernate.cfg.Configuration  - Building session factory using provided StandardServiceRegistry
[http-nio-8080-exec-8] DEBUG org.hibernate.cache.internal.RegionFactoryInitiator  - Cannot default RegionFactory based on registered strategies as `[]` RegionFactory st
[http-nio-8080-exec-8] DEBUG org.hibernate.cache.internal.RegionFactoryInitiator  - Cache region factory : org.hibernate.cache.internal.NoCachingRegionFactory
[http-nio-8080-exec-8] INFO  org.hibernate.annotations.common.Version  - HCANN000001: Hibernate Commons Annotations {5.1.2.Final}
[http-nio-8080-exec-8] DEBUG org.hibernate.type.BasicTypeRegistry  - Adding type registration boolean -> org.hibernate.type.BooleanType@651294c6
[http-nio-8080-exec-8] DEBUG org.hibernate.type.BasicTypeRegistry  - Adding type registration boolean -> org.hibernate.type.BooleanType@651294c6
[http-nio-8080-exec-8] DEBUG org.hibernate.type.BasicTypeRegistry  - Adding type registration java.lang.Boolean -> org.hibernate.type.BooleanType@651294c6
[http-nio-8080-exec-8] DEBUG org.hibernate.type.BasicTypeRegistry  - Adding type registration numeric_boolean -> org.hibernate.type.NumericBooleanType@40794e5d
[http-nio-8080-exec-8] DEBUG org.hibernate.type.BasicTypeRegistry  - Adding type registration true_false -> org.hibernate.type.TrueFalseType@38ab485a
[http-nio-8080-exec-8] DEBUG org.hibernate.type.BasicTypeRegistry  - Adding type registration yes_no -> org.hibernate.type.YesNoType@4824a9ec
[http-nio-8080-exec-8] DEBUG org.hibernate.type.BasicTypeRegistry  - Adding type registration byte -> org.hibernate.type.ByteType@364db3c9
[http-nio-8080-exec-8] DEBUG org.hibernate.type.BasicTypeRegistry  - Adding type registration byte -> org.hibernate.type.ByteType@364db3c9
[http-nio-8080-exec-8] DEBUG org.hibernate.type.BasicTypeRegistry  - Adding type registration java.lang.Byte -> org.hibernate.type.ByteType@364db3c9
[http-nio-8080-exec-8] DEBUG org.hibernate.type.BasicTypeRegistry  - Adding type registration character -> org.hibernate.type.CharacterType@14e02ddb
[http-nio-8080-exec-8] DEBUG org.hibernate.type.BasicTypeRegistry  - Adding type registration char -> org.hibernate.type.CharacterType@14e02ddb
[http-nio-8080-exec-8] DEBUG org.hibernate.type.BasicTypeRegistry  - Adding type registration java.lang.Character -> org.hibernate.type.CharacterType@14e02ddb
[http-nio-8080-exec-8] DEBUG org.hibernate.type.BasicTypeRegistry  - Adding type registration short -> org.hibernate.type.ShortType@2cac133c
[http-nio-8080-exec-8] DEBUG org.hibernate.type.BasicTypeRegistry  - Adding type registration short -> org.hibernate.type.ShortType@2cac133c
[http-nio-8080-exec-8] DEBUG org.hibernate.type.BasicTypeRegistry  - Adding type registration java.lang.Short -> org.hibernate.type.ShortType@2cac133c
[http-nio-8080-exec-8] DEBUG org.hibernate.type.BasicTypeRegistry  - Adding type registration integer -> org.hibernate.type.IntegerType@2affe5ed
[http-nio-8080-exec-8] DEBUG org.hibernate.type.BasicTypeRegistry  - Adding type registration int -> org.hibernate.type.IntegerType@2affe5ed
[http-nio-8080-exec-8] DEBUG org.hibernate.type.BasicTypeRegistry  - Adding type registration java.lang.Integer -> org.hibernate.type.IntegerType@2affe5ed
[http-nio-8080-exec-8] DEBUG org.hibernate.type.BasicTypeRegistry  - Adding type registration long -> org.hibernate.type.LongType@75d25901
[http-nio-8080-exec-8] DEBUG org.hibernate.type.BasicTypeRegistry  - Adding type registration long -> org.hibernate.type.LongType@75d25901
[http-nio-8080-exec-8] DEBUG org.hibernate.type.BasicTypeRegistry  - Adding type registration java.lang.Long -> org.hibernate.type.LongType@75d25901
[http-nio-8080-exec-8] DEBUG org.hibernate.type.BasicTypeRegistry  - Adding type registration float -> org.hibernate.type.FloatType@6fea2d10
```

Database:



```
SELECT * FROM demo.employees;
```

id	first_name	last_name	password	user_name	contactNumber	email
6	rfyj	cghh	b	a	346	ersyh
7	ram	v	dead	ram	4	r
8	rhyt	dty	gh	xfh	564	cgh
9	xdhg	xfgbf	dead	1	3453	xfgf
10	xfgvb	fgvb	fgb	fgvb	34	fgvb
11	megha	lakshmi	dead	meg	54	dfgh
12	megha	lakshmi	etg	2	54	dfgh
13	gtdh	dfgd	rfgs	dfgd	434	srgs
14	dth	dth	dtdh	dfhf	45	dhtd
15	dxf	frt	rt	rty	54	rty
16	zrxdtgh	cftg	xfrgth	fxgy	3453	fgyf
17	Rathi	Meena	dead	Rat	35636	xfghdf
18	rfghrf	rth	rfgthrf	rftghfr	3563	rfghf
19	nagesh	vaith	dead	nag	3526	sthfdht
20	ffffffffffffffffffffff...	xfbcxfb	cfgbc	cgbbg	444444444444	gbcfgb
21	dxg	df	dxfvdx	dxfx	34	fdbvd
22	dhdc	cdgbcf	gfnfv	cfgcf	45	fcg
23	dgh	xfgh	fxg	xfg	3453246	xdf
24	xfgh	fcgbh	dcfgb	fcgh	43	dcf
25	xgs	xfbx	xdf	xfbx	346	xsdgv

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

No object selected

Query to create database and schema:

Create database demo;

CREATE TABLE drinks (

   id int NOT NULL PRIMARY KEY,

   first_name varchar(255),

   last_name varchar(255),

   user_name varchar(255),

   password varchar(255),

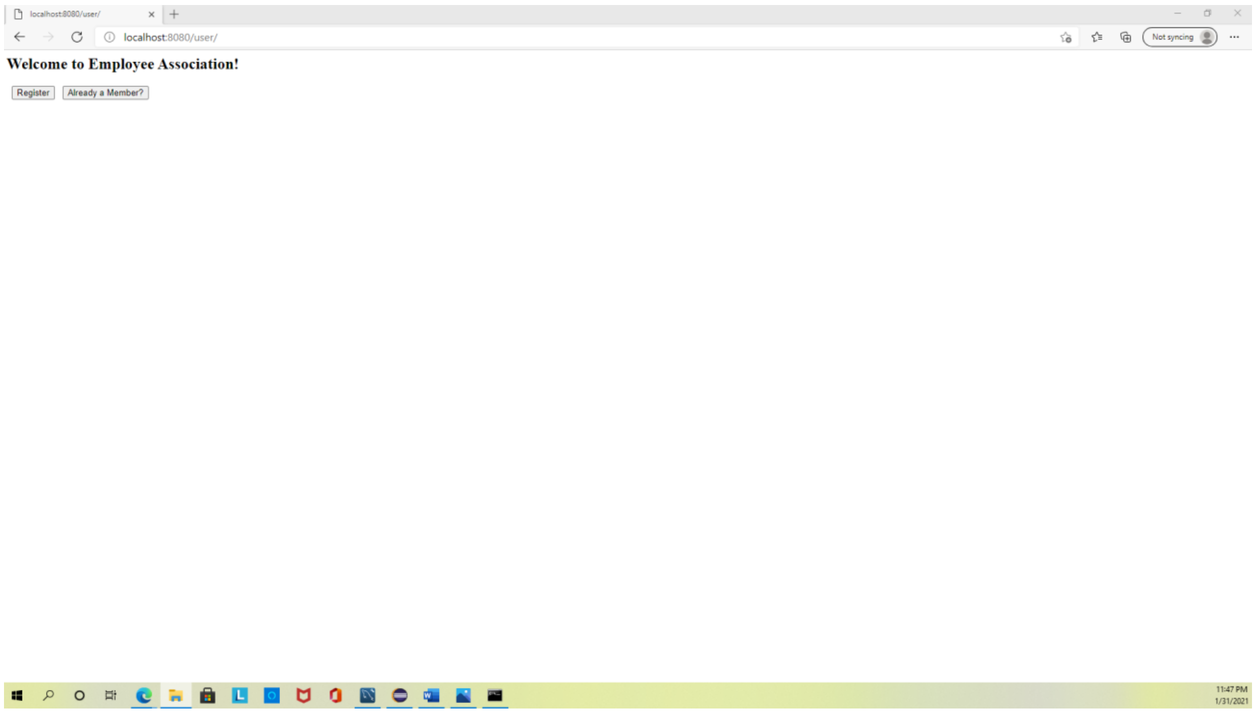
   contact_number bigint,

   email varchar(255)

);

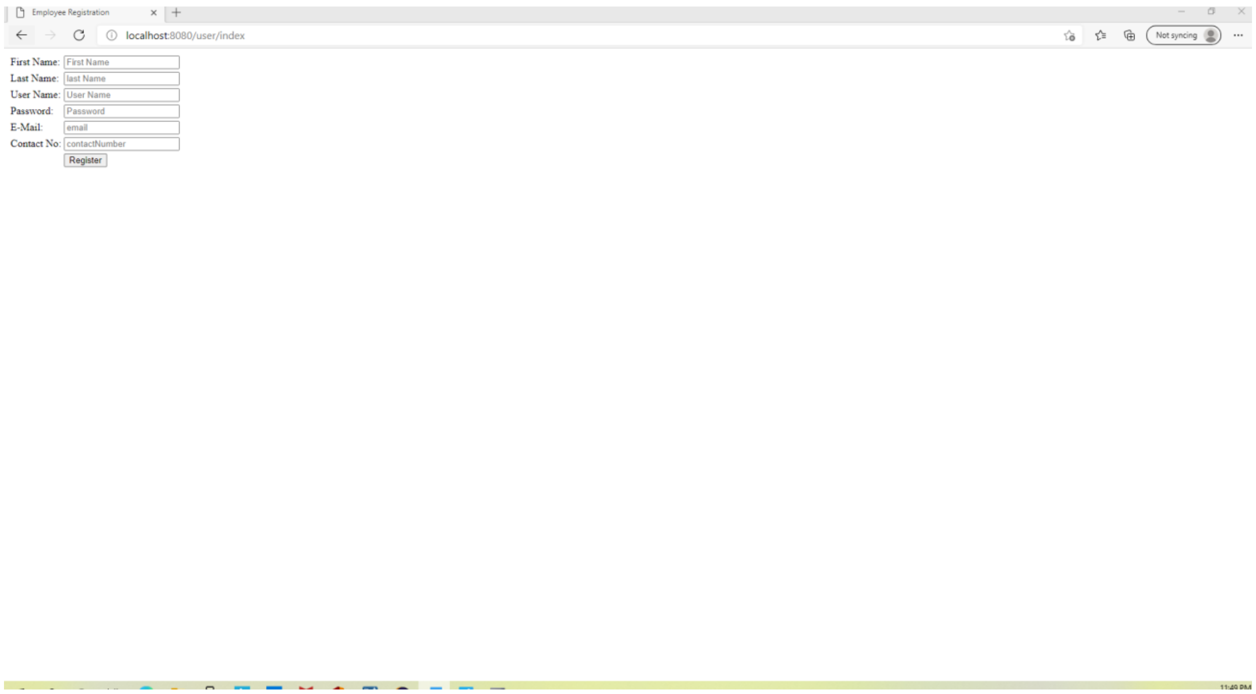
Outputs:

Welcome Page


localhost:8080/user/

Welcome to Employee Association!

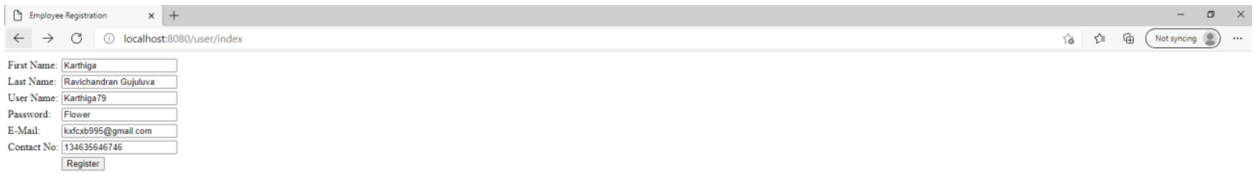
Registration Empty page


Employee Registration

localhost:8080/user/index

First Name:	First Name
Last Name:	last Name
User Name:	User Name
Password:	Password
E-Mail:	email
Contact No:	contactNumber

Registration with details page

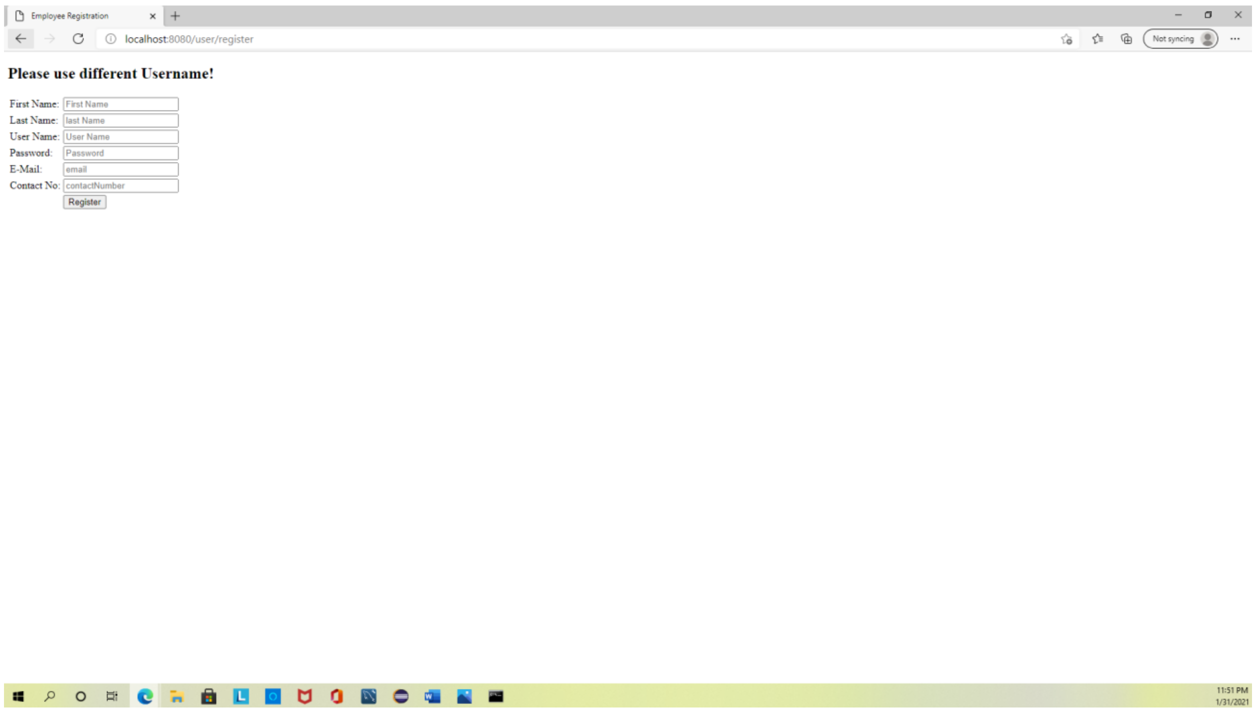


localhost:8080/user/index

First Name:	Karthiga
Last Name:	Ravichandran Gujuluva
User Name:	Karthiga79
Password:	Flower
E-Mail:	kxfcxb995@gmail.com
Contact No:	134635646746

Scenario 1:

Registration page asking for different username when given same username



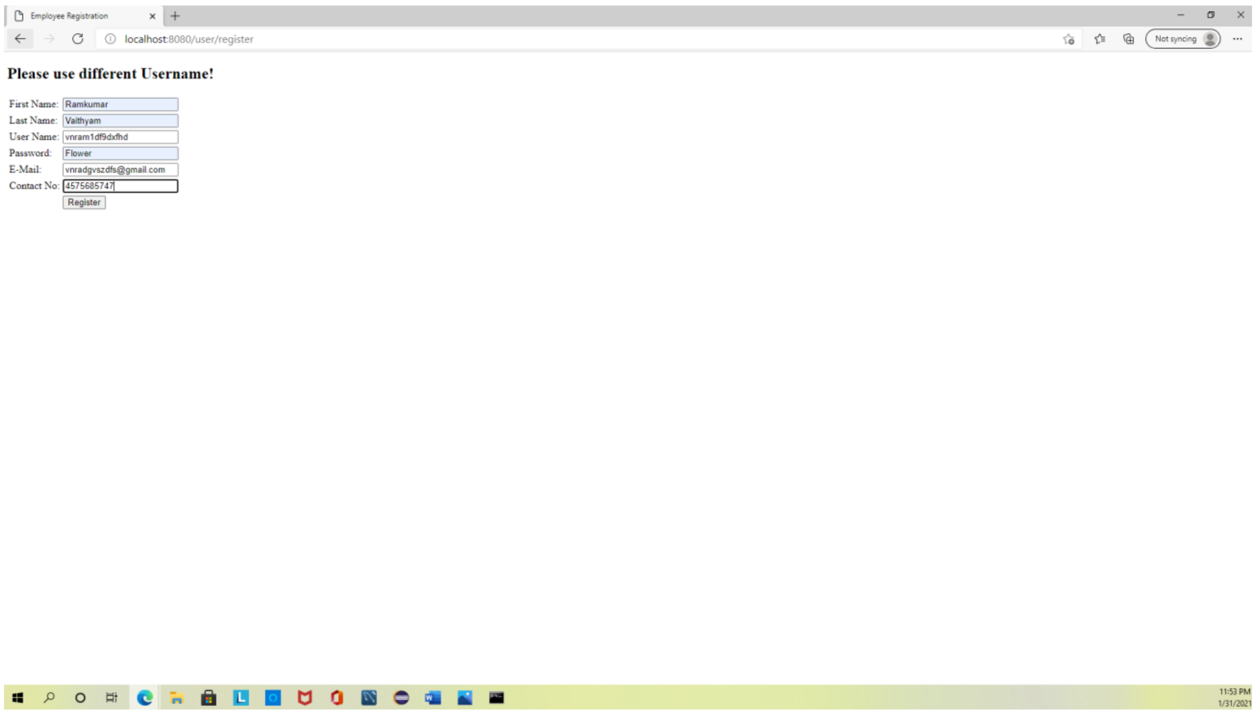
localhost:8080/user/register

Please use different Username!

First Name:	First Name
Last Name:	last Name
User Name:	User Name
Password:	Password
E-Mail:	email
Contact No:	contactNumber

Registration page with different Details and error

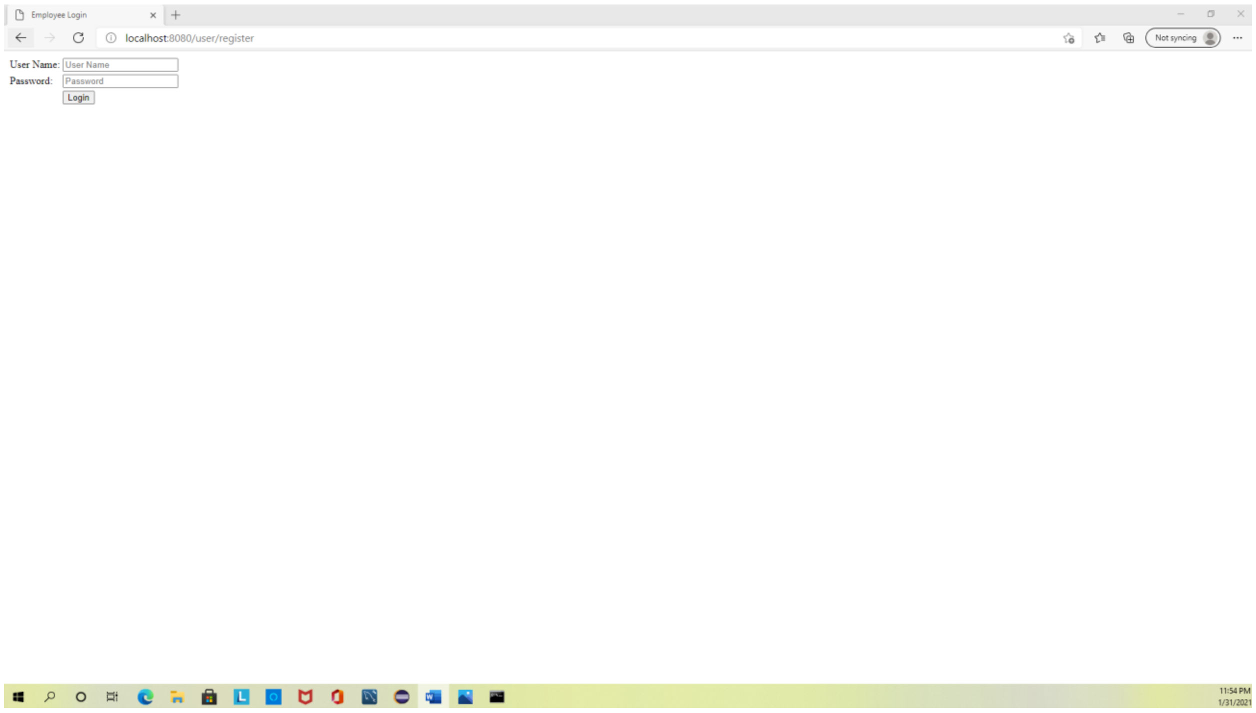
Scenario 2:



Please use different Username!

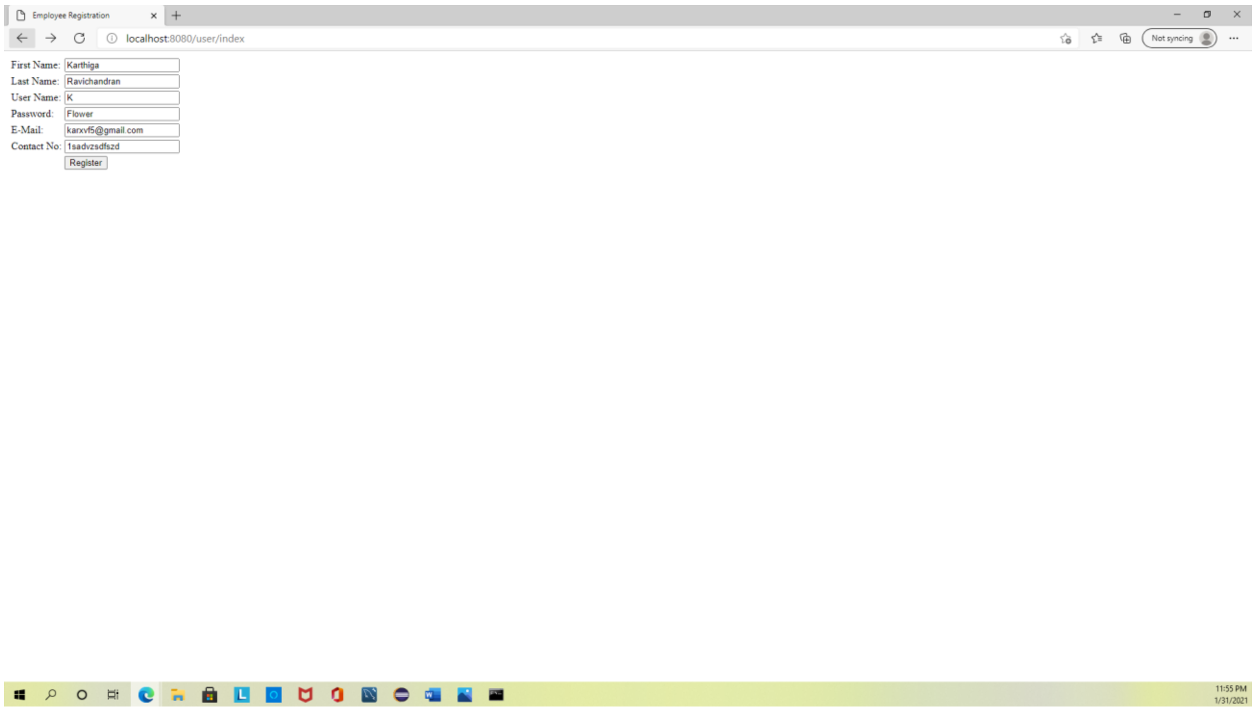
First Name:	Ramkumar
Last Name:	Vaithyam
User Name:	vnram1df9dxfhd
Password:	Flower
E-Mail:	vnradgvszdfs@gmail.com
Contact No:	4575685747

Successful Registration leading to login page



Scenario 2:

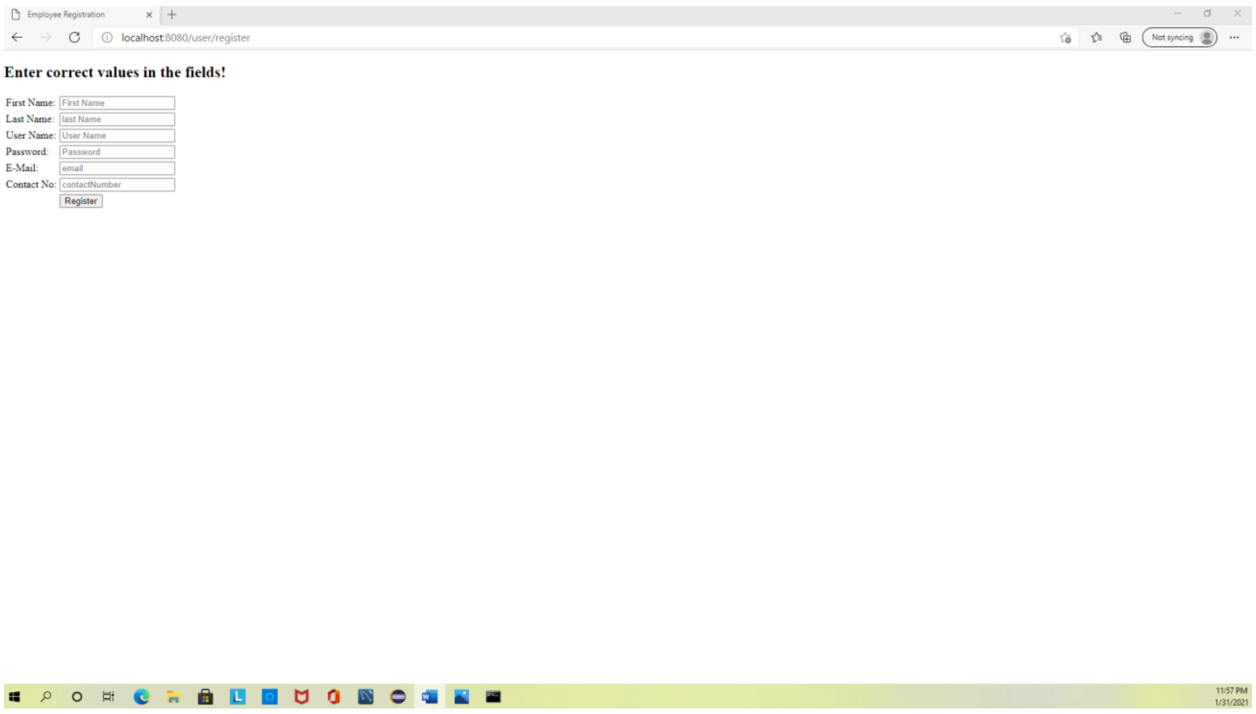
Registration with incorrect phone number with strings


Employee Registration

localhost:8080/user/index

First Name:	Karthiga
Last Name:	Ravichandran
User Name:	K
Password:	Flower
E-Mail:	karxvf5@gmail.com
Contact No:	1sadvzsdfszd

Registration Error - Incorrect Values Error

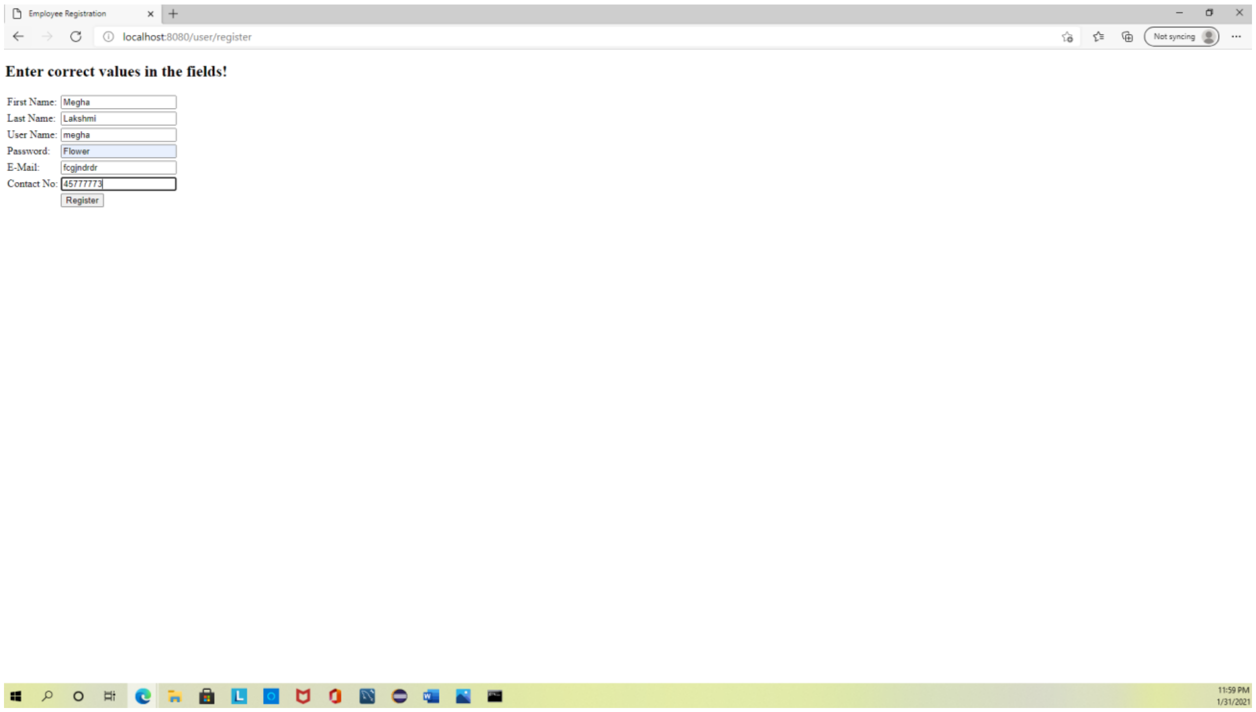

Employee Registration

localhost:8080/user/register

Enter correct values in the fields!

First Name:	First Name
Last Name:	last Name
User Name:	User Name
Password:	Password
E-Mail:	email
Contact No:	contactNumber

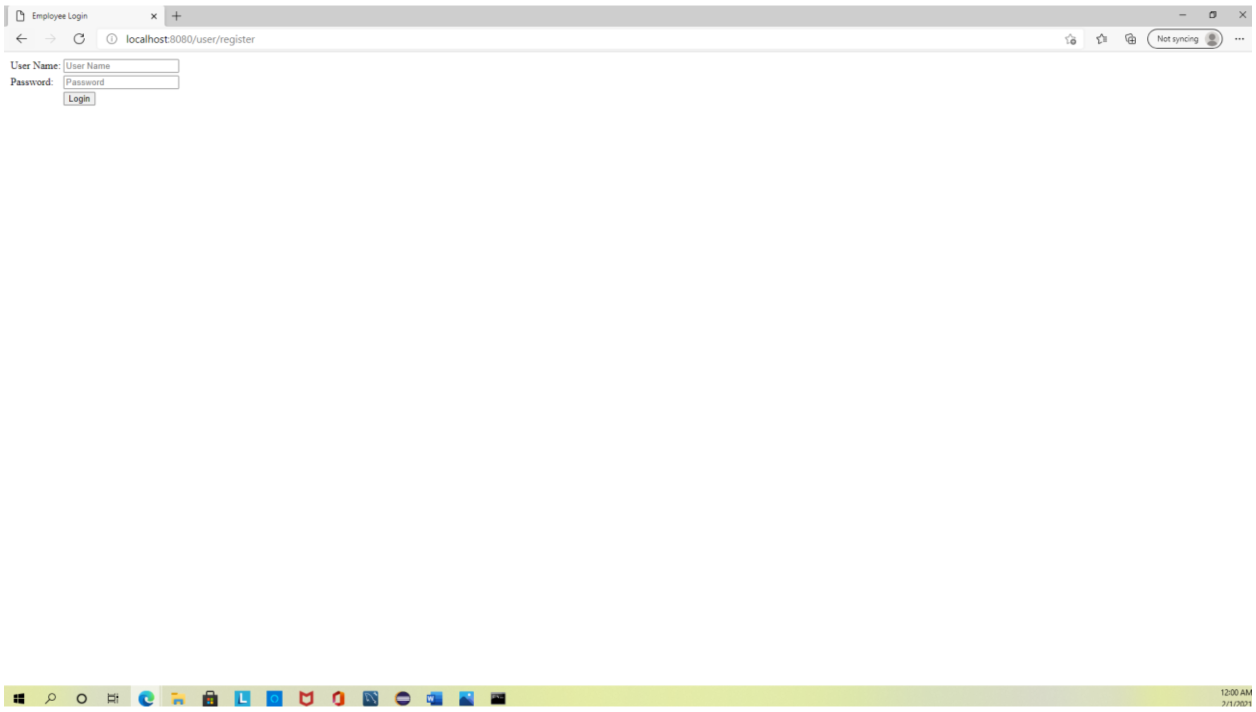
Registration page with correct values



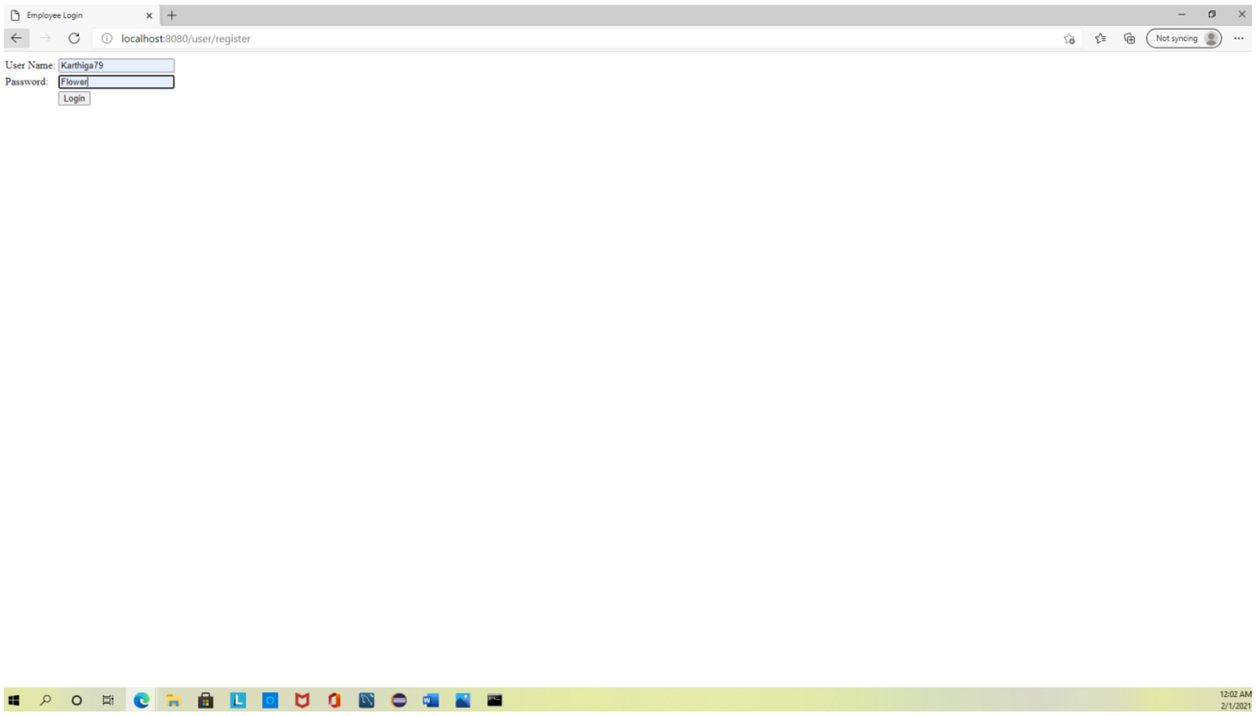
Enter correct values in the fields!

First Name:	Megha
Last Name:	Lakshmi
User Name:	megha
Password:	Flower
E-Mail:	fcgjndrdr
Contact No:	45777773

Successful registration leading to Login Page



Correct Values Login Page



Employee Login

localhost:8080/user/register

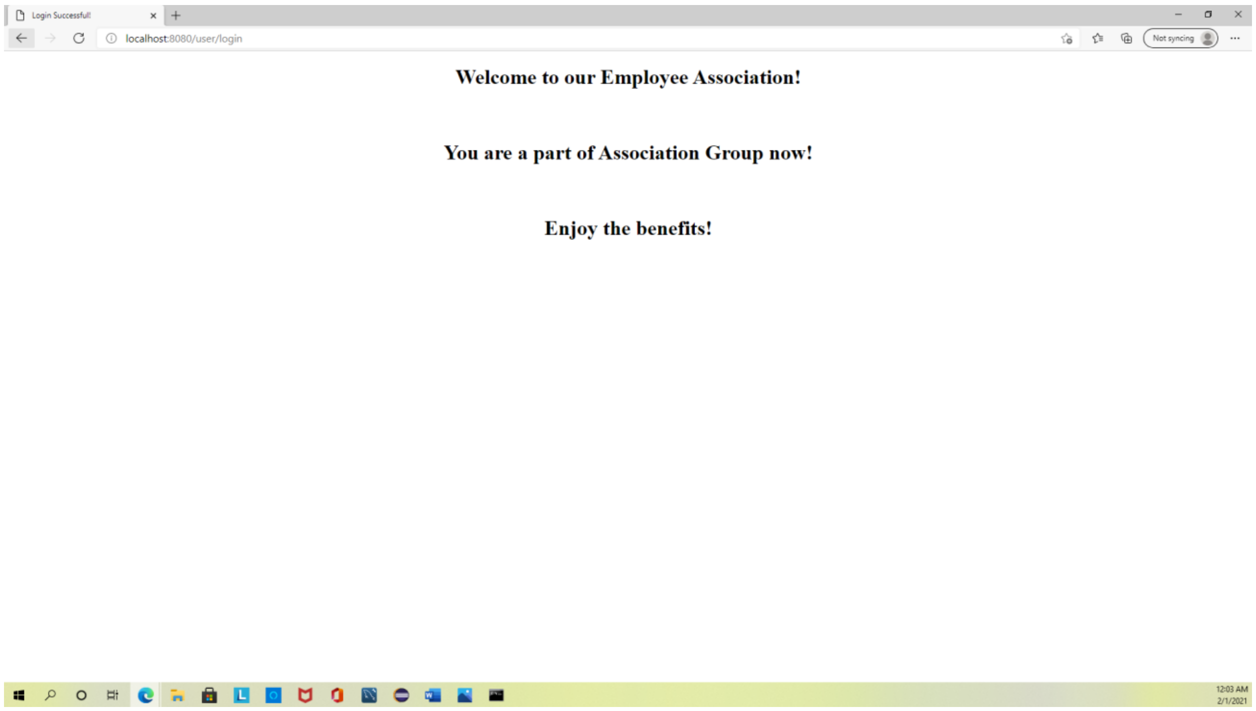
User Name: Karthiga79

Password: Flower

Login

12:02 AM 2/1/2021

Successful Login



Login Successfull

localhost:8080/user/login

Welcome to our Employee Association!

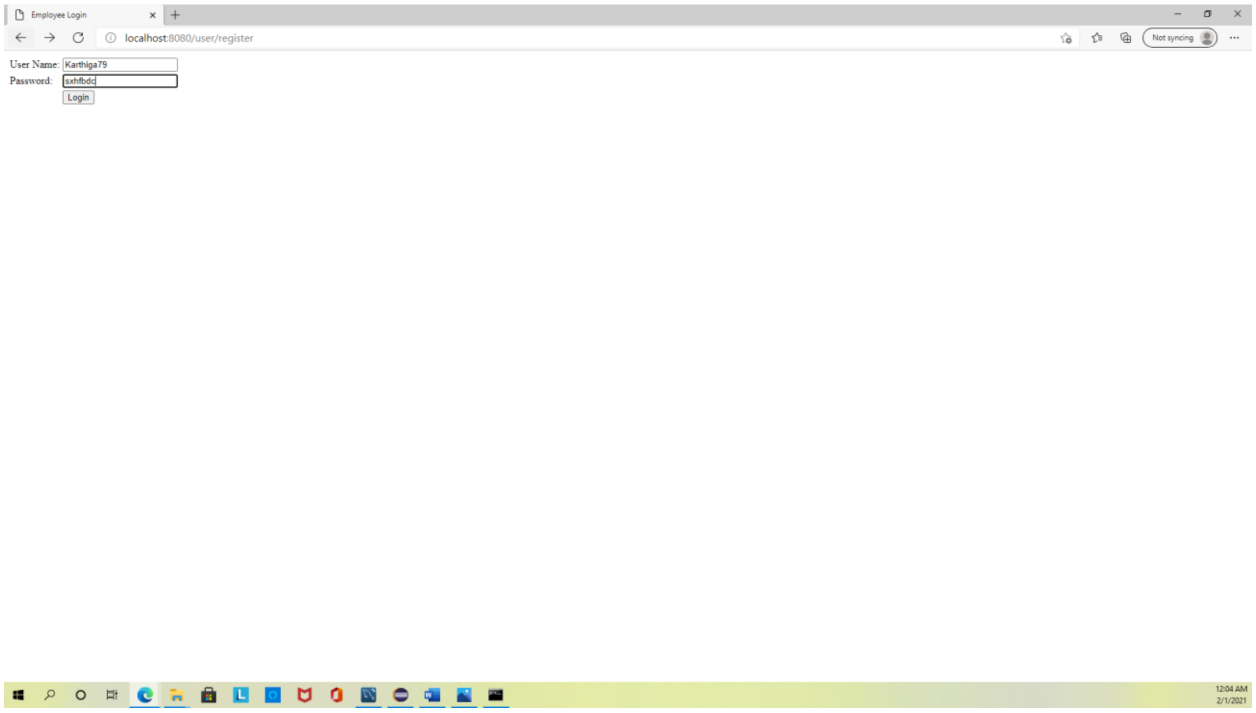
You are a part of Association Group now!

Enjoy the benefits!

12:03 AM 2/1/2021

Scenario 1:

Wrong Password Login Page


Employee Login

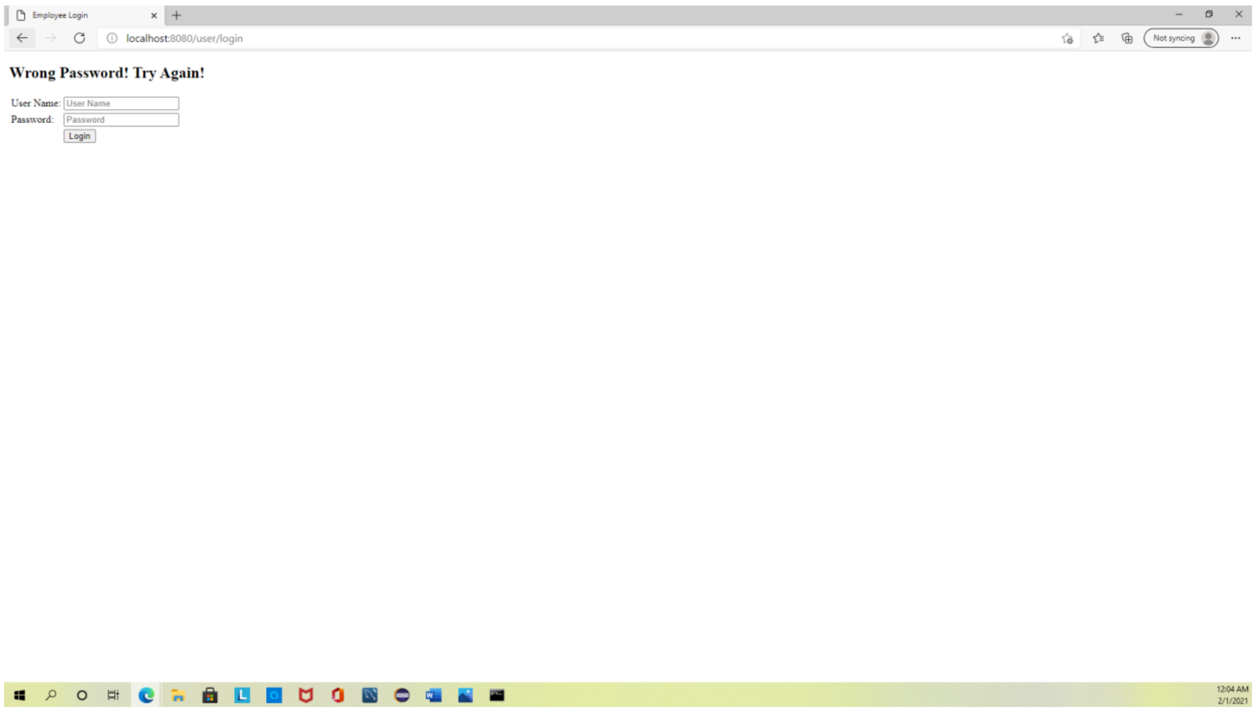
localhost:8080/user/register

User Name: Karthiga79

Password: sxhfbdc

Login

Wrong Password Error Message Login Page


Employee Login

localhost:8080/user/login

Wrong Password! Try Again!

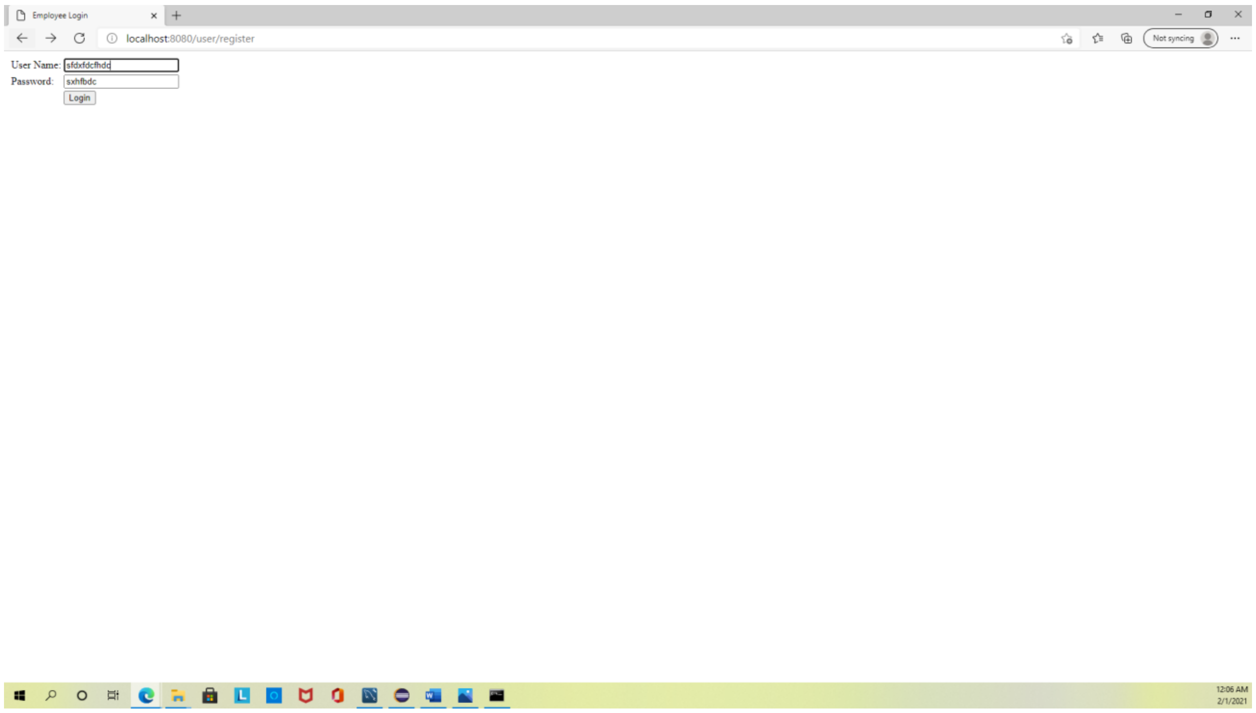
User Name: User Name

Password: Password

Login

Scenario 2:

Incorrect username login page or username that does not exist in DB

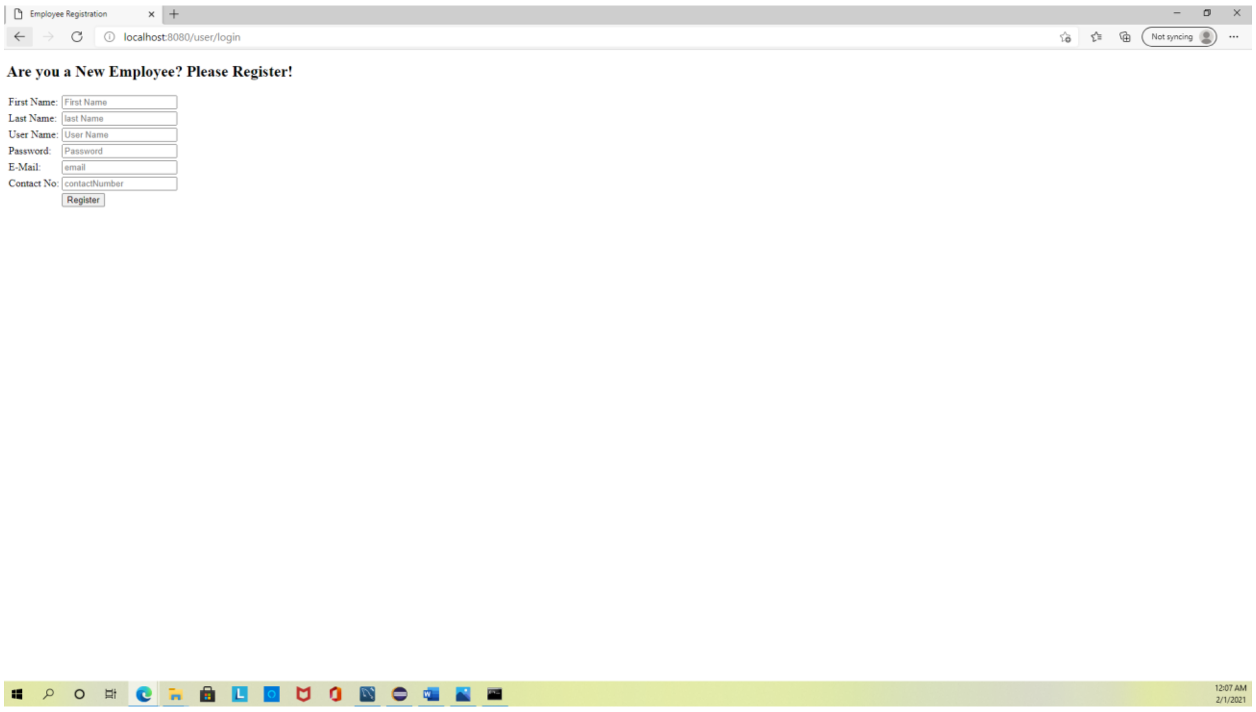

Employee Login

localhost:8080/user/register

User Name: sfdxfdcfhdd
Password: sxhfbdc
Login

12:06 AM
2/1/2021

Login Page leading to Registration Page with Missing user error message


Employee Registration

localhost:8080/user/login

Are you a New Employee? Please Register!

First Name: First Name
Last Name: last Name
User Name: User Name
Password: Password
E-Mail: email
Contact No: contactNumber
Register

12:07 AM
2/1/2021

Code:

Employee Dao.java

package com.hcl.dao;

```
import org.apache.log4j.Logger;
import org.hibernate.Session;
import org.hibernate.Transaction;

import com.hcl.model.Employee;
import com.hcl.util.HibernateUtil;

public class EmployeeDao {
	static Logger log = Logger.getLogger(EmployeeDao.class.getName());
	public boolean saveUser(Employee employee) {
		Transaction transaction = null;
		try (Session session = HibernateUtil.getSessionFactory().openSession()) {
			transaction = session.beginTransaction();
			session.save(employee);
			//hibernate insert query to save user details in DB
			transaction.commit();
			log.info("Data is saved");
			return true;
		} catch (Exception e) {
			e.printStackTrace();
			return false;
		}
	}

	public String validate(String username, String password) {
		Transaction transaction = null;
		Employee user = null;
		try (Session session = HibernateUtil.getSessionFactory().openSession()) {
```

```
                        transaction = session.beginTransaction();

                        //To find if employee exists in DB

                        user = findIfEmployeeExists(username);

                        //If user exist, go to login page

                        if (user != null) {

                                if (user.getPassword().equals(password)) {

                                        log.info("Pass");

                                        return "Pass";

                                } else {

                                        //If user exist but gave wrong password, ask again to give
correct password

                                        log.info("Exists");

                                        return "Exists";

                                }

                        } else {

                                //If user doesn't exist, ask to register

                                log.info("NotExists");

                                return "NotExists";

                        }

                } catch (Exception e) {

                        e.printStackTrace();

                }

                log.info("TryAgain");

                return "TryAgain";

        }

//To find if employee exists in DB

        public Employee findIfEmployeeExists(String username) {

                Transaction transaction = null;

                Employee employee = null;
```

```
		try (Session session = HibernateUtil.getSessionFactory().openSession()) {
			transaction = session.beginTransaction();
			//hibernate select query to find if user exists in DB
			employee = (Employee) session.createQuery("FROM Employee U WHERE
U.username = :username")
					.setParameter("username", username).uniqueResult();
			log.info("Finding if data exists");
		} catch (Exception e) {
			e.printStackTrace();
		}
		return employee;
	}
}
```

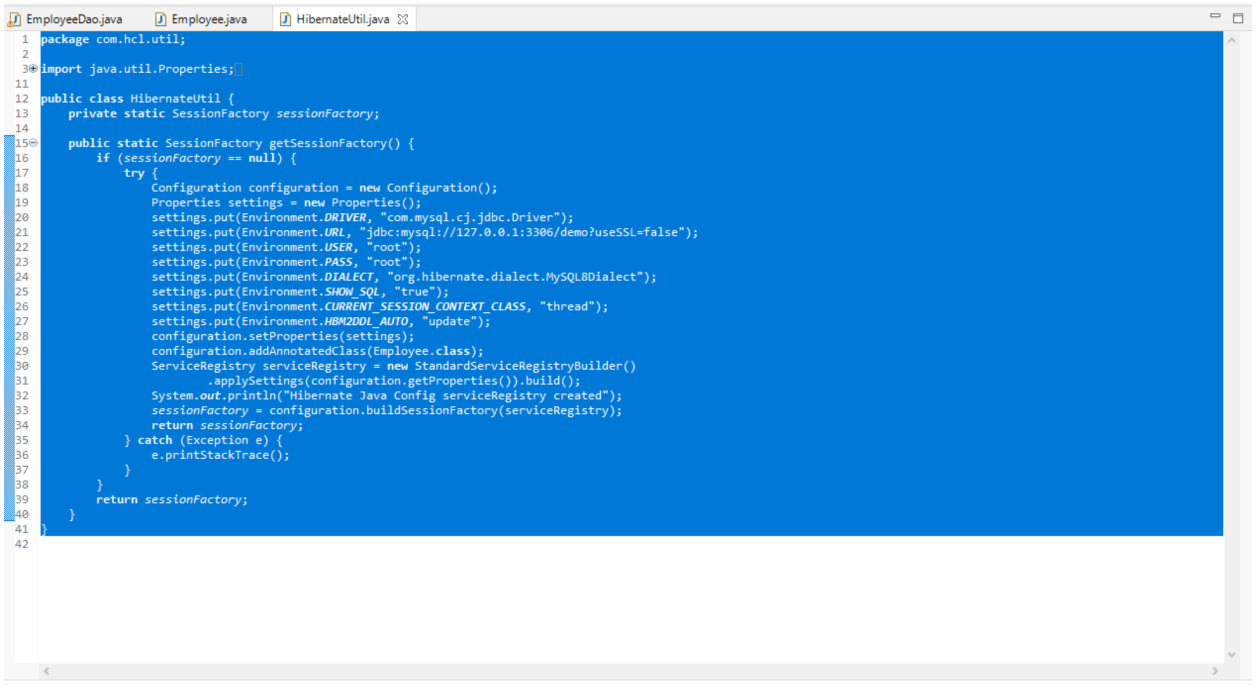
Employee.java

```
package com.hcl.model;


import java.io.Serializable;


import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;


@Entity
@Table(name = "employees")
public class Employee implements Serializable {
	private static final long serialVersionUID = 1L;
}

@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private int id;

@Column(name = "first_name")
private String firstName;

@Column(name = "last_name")
private String lastName;

@Column(name = "user_name")
private String username;

@Column(name = "password")
private String password;

@Column(name = "email")
private String email;

@Column(name = "contactNumber")
private String contactNumber;

/**
 * @return the email
 */
public String getEmail() {
		return email;
}
```

```
/**
 * @param email the email to set
 */
public void setEmail(String email) {
	this.email = email;
}

/**
 * @return the contactNumber
 */
public String getContactNumber() {
	return contactNumber;
}

/**
 * @param contactNumber the contactNumber to set
 */
public void setContactNumber(String contactNumber) {
	this.contactNumber = contactNumber;
}

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;
	}
}
```

HibernateUtil.java


EmployeeDao.java Employee.java HibernateUtil.java

```java
package com.hcl.util;

import java.util.Properties;

public class HibernateUtil {
    private static SessionFactory sessionFactory;

    public static SessionFactory getSessionFactory() {
        if (sessionFactory == null) {
            try {
                Configuration configuration = new Configuration();
                Properties settings = new Properties();
                settings.put(Environment.DRIVER, "com.mysql.cj.jdbc.Driver");
                settings.put(Environment.URL, "jdbc:mysql://127.0.0.1:3306/demo?useSSL=false");
                settings.put(Environment.USER, "root");
                settings.put(Environment.PASS, "root");
                settings.put(Environment.DIALECT, "org.hibernate.dialect.MySQL8Dialect");
                settings.put(Environment.SHOW_SQL, "true");
                settings.put(Environment.CURRENT_SESSION_CONTEXT_CLASS, "thread");
                settings.put(Environment.HBM2DDL_AUTO, "update");
                configuration.setProperties(settings);
                configuration.addAnnotatedClass(Employee.class);
                ServiceRegistry serviceRegistry = new StandardServiceRegistryBuilder()
                        .applySettings(configuration.getProperties()).build();
                System.out.println("Hibernate Java Config serviceRegistry created");
                sessionFactory = configuration.buildSessionFactory(serviceRegistry);
                return sessionFactory;
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        return sessionFactory;
    }
}
```

EmployeeLoginServlet.java

```java
package com.hcl.web;


import java.io.IOException;


import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;


import org.apache.log4j.Logger;


import com.hcl.dao.EmployeeDao;
```

```
@WebServlet("/login")
public class EmployeeLoginServlet extends HttpServlet {
	static Logger log = Logger.getLogger(EmployeeDao.class.getName());
	private static final long serialVersionUID = 1L;
	private EmployeeDao loginDao;

	public EmployeeLoginServlet() {
		super();
		loginDao = new EmployeeDao();
	}

	protected void doPost(HttpServletRequest request, HttpServletResponse response)
			throws ServletException, IOException {
		String message = null;
		try {
			authenticate(request, response);
		} catch (Exception e) {
			//If unexpected error occured, ask to login again
			RequestDispatcher dispatcher = request.getRequestDispatcher("login.jsp");
			message = "Unexpected Error Occured! Try again!";
			//sending message to user
			log.info("Unexpected Error Occured! Try again!");
			request.setAttribute("message", message);
			dispatcher.forward(request, response);
		}
	}

	private void authenticate(HttpServletRequest request, HttpServletResponse response) throws
Exception {
		String username = request.getParameter("username");
		String password = request.getParameter("password");
		String message = null;
		String check = loginDao.validate(username, password);
		if (check.equals("Pass")) {
			//If user exist, Successfull login
			System.out.println("LoginSuccess");
			log.info("LoginSuccess");
			RequestDispatcher dispatcher =
request.getRequestDispatcher("SuccessLogin.jsp");
			dispatcher.forward(request, response);
		} else if (check.equals("Exists")) {
			//If user exist but gave wrong password, ask again to give correct password
			System.out.println("Exists");
			log.info("Exists");
			RequestDispatcher dispatcher = request.getRequestDispatcher("login.jsp");
			message = "Wrong Password! Try Again!";
			//sending message to user
			request.setAttribute("message", message);
			dispatcher.forward(request, response);
		} else if (check.equals("NotExists")) {
			//If user doesn't exist, ask to register
			System.out.println("NotExists");
			log.info("NotExists");
			RequestDispatcher dispatcher = request.getRequestDispatcher("register.jsp");
			message = "Are you a New Employee? Please Register!";
			//sending message to user
			request.setAttribute("message", message);
			dispatcher.forward(request, response);
			} else {
				//If unexpected error occured, ask to login again
				System.out.println("Unexpected");
				log.info("Unexpected Error Occured! Try again!");
				RequestDispatcher dispatcher = request.getRequestDispatcher("login.jsp");
				message = "Unexpected Error Occured! Try again!";
				//sending message to user
				request.setAttribute("message", message);
				dispatcher.forward(request, response);
			}
		}
}
```

EmployeeRegistrationServlet.java

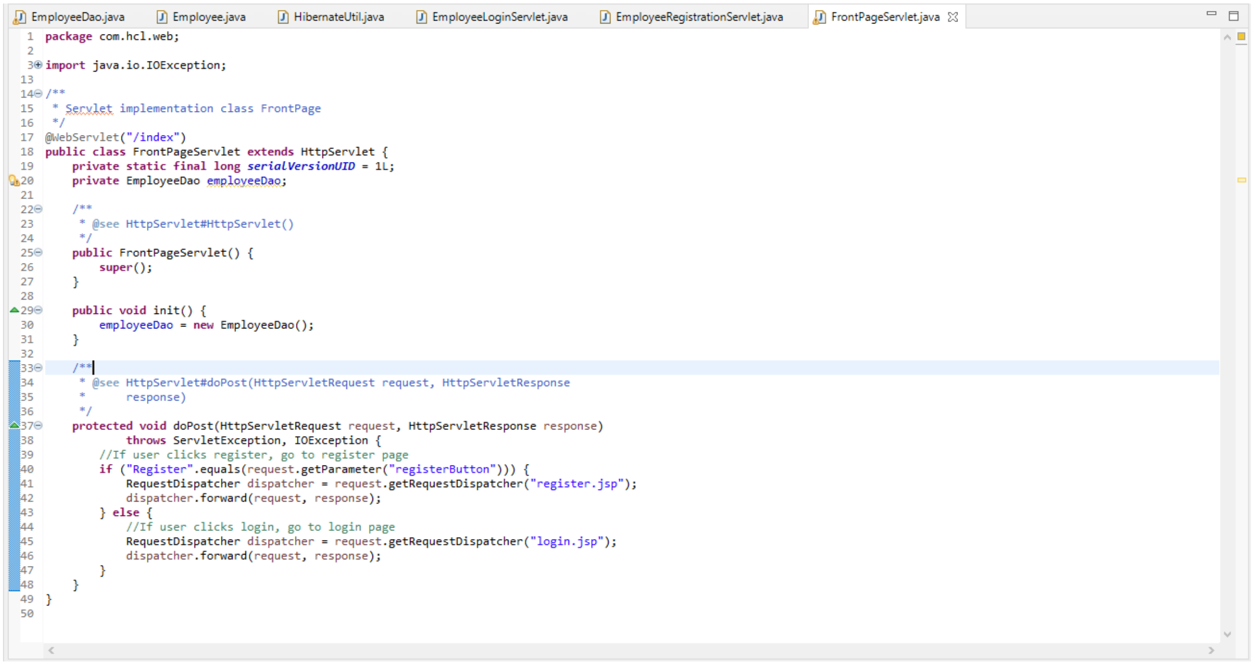
```
package com.hcl.web;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.log4j.Logger;

import com.hcl.dao.EmployeeDao;
import com.hcl.model.Employee;
```

```
@WebServlet("/register")
public class EmployeeRegistrationServlet extends HttpServlet {
	static Logger log = Logger.getLogger(EmployeeDao.class.getName());
	private static final long serialVersionUID = 1L;
	private EmployeeDao employeeDao;

	protected void doPost(HttpServletRequest request, HttpServletResponse response)
			throws ServletException, IOException {
		register(request, response);
	}

	private void register(HttpServletRequest request, HttpServletResponse response)
			throws IOException, ServletException {
		String message = null;
		Employee ifEmployeeExists = 
employeeDao.findIfEmployeeExists(request.getParameter("username"));
		if (ifEmployeeExists == null) {
			System.out.println("Saved");
			//get all values entered by the user
			String firstName = request.getParameter("firstName");
			String lastName = request.getParameter("lastName");
			String username = request.getParameter("username");
			String password = request.getParameter("password");
			String email = request.getParameter("email");
			String contactNumber = request.getParameter("contactNumber");
			Employee employee = new Employee();
			//store the values in employee
			employee.setFirstName(firstName);
			employee.setLastName(lastName);
```

```
						employee.setUsername(username);
						employee.setPassword(password);
						employee.setEmail(email);
						employee.setContactNumber(contactNumber);
						if (employeeDao.saveUser(employee)) {
							RequestDispatcher dispatcher = 
request.getRequestDispatcher("login.jsp");
							dispatcher.forward(request, response);
						} else {
							RequestDispatcher dispatcher = 
request.getRequestDispatcher("register.jsp");
							message = "Enter correct values in the fields!";
							//sending message to user
							request.setAttribute("message", message);
							log.info(message);
							dispatcher.forward(request, response);
						}
					} else {
						System.out.println("Same User");
						RequestDispatcher dispatcher = request.getRequestDispatcher("register.jsp");
						//sending message to user
						message = "Please use different Username!";
						request.setAttribute("message", message);
						log.info(message);
						dispatcher.forward(request, response);
					}
				}

				public EmployeeRegistrationServlet() {
```

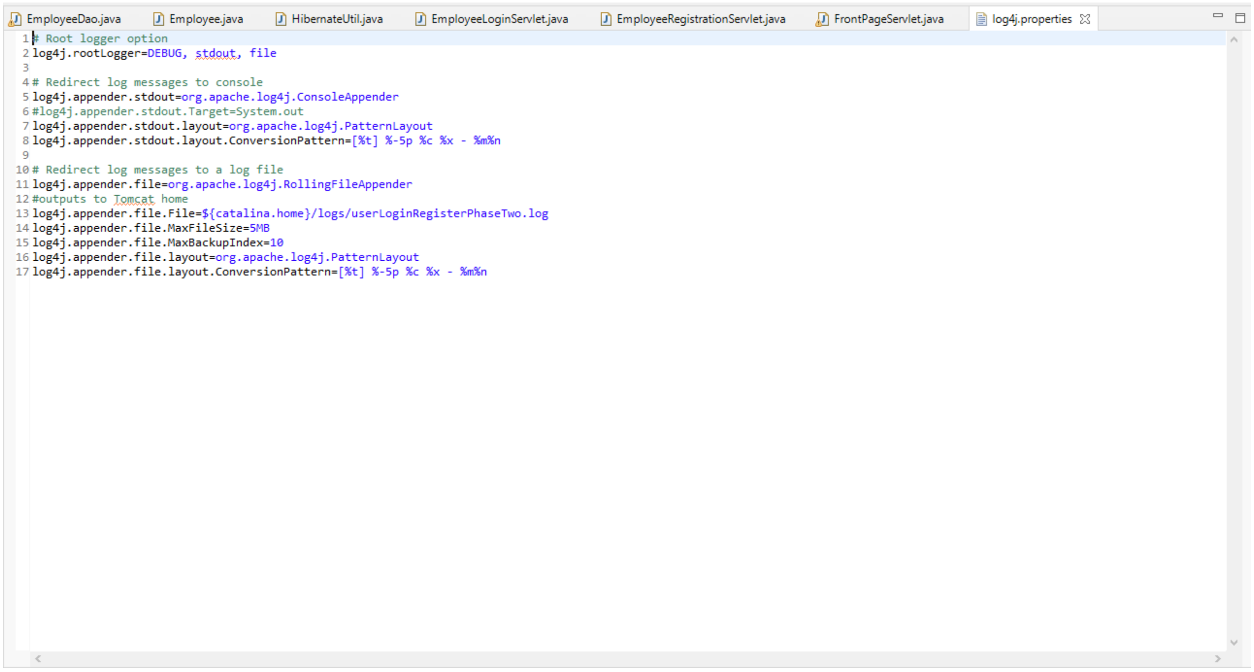
```
                super();

                employeeDao = new EmployeeDao();

        }

}
```

FrontPageServlet.java



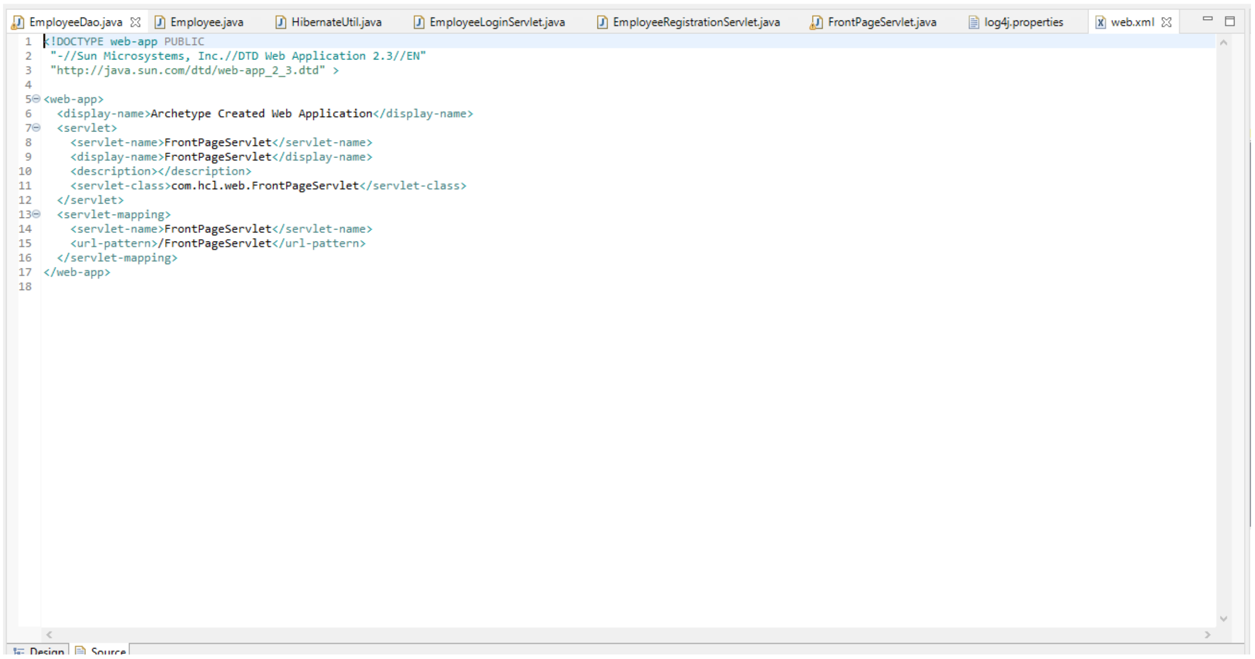
```java
package com.hcl.web;

import java.io.IOException;

/**
 * Servlet implementation class FrontPage
 */
@WebServlet("/index")
public class FrontPageServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private EmployeeDao employeeDao;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public FrontPageServlet() {
        super();
    }

    public void init() {
        employeeDao = new EmployeeDao();
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
     *      response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        //If user clicks register, go to register page
        if ("Register".equals(request.getParameter("registerButton"))) {
            RequestDispatcher dispatcher = request.getRequestDispatcher("register.jsp");
            dispatcher.forward(request, response);
        } else {
            //If user clicks login, go to login page
            RequestDispatcher dispatcher = request.getRequestDispatcher("login.jsp");
            dispatcher.forward(request, response);
        }
    }
}
```

Log4j.properties



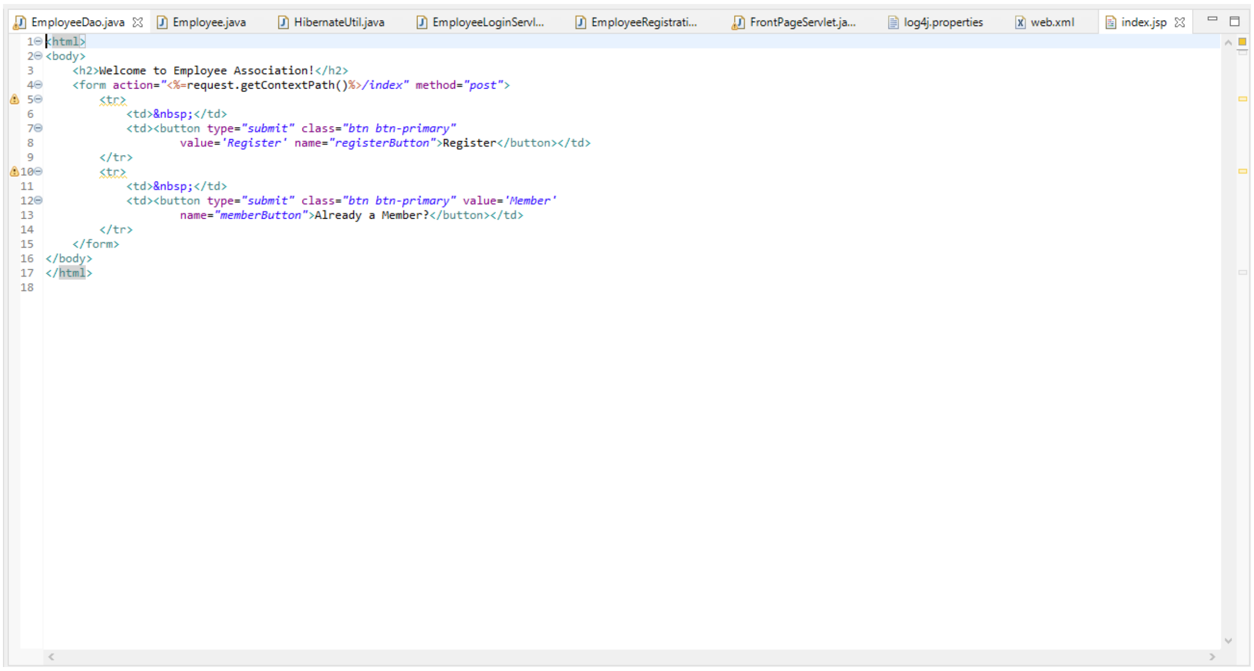
```
# Root logger option
log4j.rootLogger=DEBUG, stdout, file

# Redirect log messages to console
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
#log4j.appender.stdout.Target=System.out
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=[%t] %-5p %c %x - %m%n

# Redirect log messages to a log file
log4j.appender.file=org.apache.log4j.RollingFileAppender
#outputs to Tomcat home
log4j.appender.file.File=${catalina.home}/logs/userLoginRegisterPhaseTwo.log
log4j.appender.file.MaxFileSize=5MB
log4j.appender.file.MaxBackupIndex=10
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=[%t] %-5p %c %x - %m%n
```

Web-xml



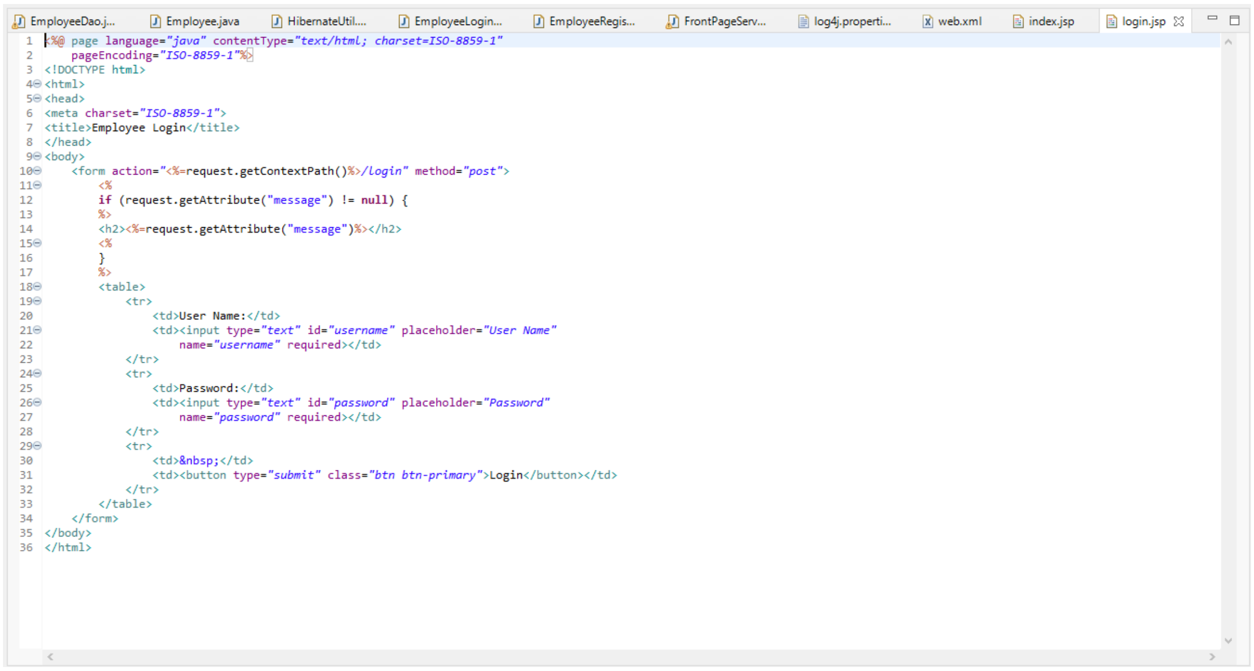
```xml
<!DOCTYPE web-app PUBLIC
 "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
 "http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>
  <display-name>Archetype Created Web Application</display-name>
  <servlet>
    <servlet-name>FrontPageServlet</servlet-name>
    <display-name>FrontPageServlet</display-name>
    <description></description>
    <servlet-class>com.hcl.web.FrontPageServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>FrontPageServlet</servlet-name>
    <url-pattern>/FrontPageServlet</url-pattern>
  </servlet-mapping>
</web-app>
```

Index.jsp



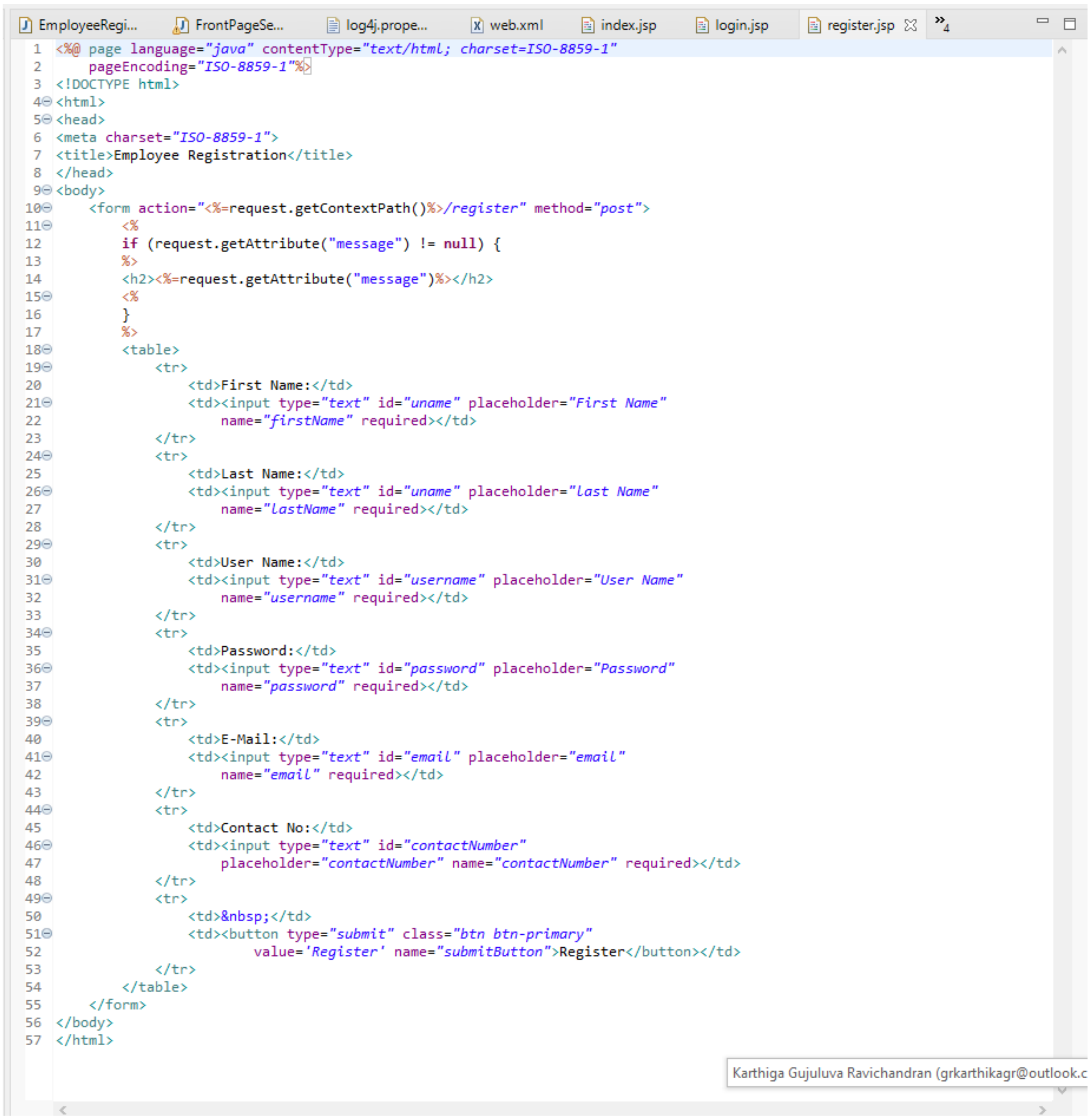
```html
<html>
<body>
    <h2>Welcome to Employee Association!</h2>
    <form action="<%=request.getContextPath()%>/index" method="post">
        <tr>
            <td>&nbsp;</td>
            <td><button type="submit" class="btn btn-primary"
                    value='Register' name="registerButton">Register</button></td>
        </tr>
        <tr>
            <td>&nbsp;</td>
            <td><button type="submit" class="btn btn-primary" value='Member'
                    name="memberButton">Already a Member?</button></td>
        </tr>
    </form>
</body>
</html>
```

Login.jsp



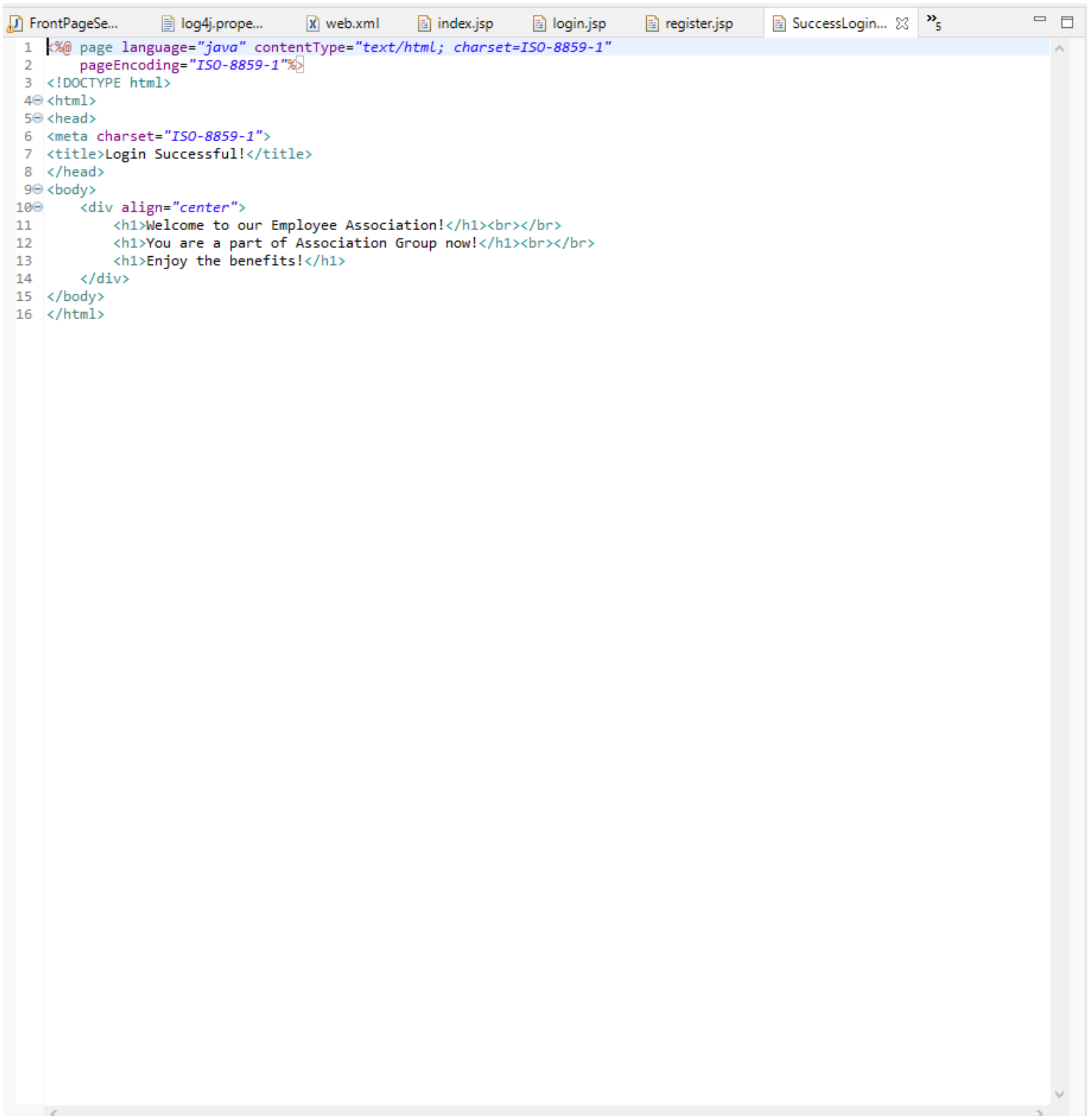
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Employee Login</title>
</head>
<body>
    <form action="<%=request.getContextPath()%>/login" method="post">
        <%
        if (request.getAttribute("message") != null) {
        %>
        <h2><%=request.getAttribute("message")%></h2>
        <%
        }
        %>
        <table>
            <tr>
                <td>User Name:</td>
                <td><input type="text" id="username" placeholder="User Name"
                    name="username" required></td>
            </tr>
            <tr>
                <td>Password:</td>
                <td><input type="text" id="password" placeholder="Password"
                    name="password" required></td>
            </tr>
            <tr>
                <td>&nbsp;</td>
                <td><button type="submit" class="btn btn-primary">Login</button></td>
            </tr>
        </table>
    </form>
</body>
</html>
```

Register.jsp



```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Employee Registration</title>
</head>
<body>
    <form action="<%=request.getContextPath()%>/register" method="post">
        <%
        if (request.getAttribute("message") != null) {
        %>
        <h2><%=request.getAttribute("message")%></h2>
        <%
        }
        %>
        <table>
            <tr>
                <td>First Name:</td>
                <td><input type="text" id="uname" placeholder="First Name"
                    name="firstName" required></td>
            </tr>
            <tr>
                <td>Last Name:</td>
                <td><input type="text" id="uname" placeholder="Last Name"
                    name="lastName" required></td>
            </tr>
            <tr>
                <td>User Name:</td>
                <td><input type="text" id="username" placeholder="User Name"
                    name="username" required></td>
            </tr>
            <tr>
                <td>Password:</td>
                <td><input type="text" id="password" placeholder="Password"
                    name="password" required></td>
            </tr>
            <tr>
                <td>E-Mail:</td>
                <td><input type="text" id="email" placeholder="email"
                    name="email" required></td>
            </tr>
            <tr>
                <td>Contact No:</td>
                <td><input type="text" id="contactNumber"
                    placeholder="contactNumber" name="contactNumber" required></td>
            </tr>
            <tr>
                <td>&nbsp;</td>
                <td><button type="submit" class="btn btn-primary"
                        value='Register' name="submitButton">Register</button></td>
            </tr>
        </table>
    </form>
</body>
</html>
```

SuccessLogin.jsp



```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Login Successful!</title>
</head>
<body>
    <div align="center">
        <h1>Welcome to our Employee Association!</h1><br></br>
        <h1>You are a part of Association Group now!</h1><br></br>
        <h1>Enjoy the benefits!</h1>
    </div>
</body>
</html>
```

Pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">

       <modelVersion>4.0.0</modelVersion>
```

```xml
<groupId>com.hcl</groupId>
<artifactId>userLoginRegisterPhaseTwo</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>war</packaging>

<name>userLoginRegisterPhaseTwo Maven Webapp</name>

<properties>
	<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
	<maven.compiler.source>1.8</maven.compiler.source>
	<maven.compiler.target>1.8</maven.compiler.target>
</properties>

<dependencies>
	<dependency>
		<groupId>javax.servlet</groupId>
		<artifactId>javax.servlet-api</artifactId>
		<version>4.0.1</version>
		<scope>provided</scope>
	</dependency>
	<dependency>
		<groupId>javax.servlet.jsp</groupId>
		<artifactId>javax.servlet.jsp-api</artifactId>
		<version>2.3.3</version>
		<scope>provided</scope>
	</dependency>
	<dependency>
		<groupId>org.hibernate</groupId>
		<artifactId>hibernate-core</artifactId>
		<version>5.4.27.Final</version>
	</dependency>
	<dependency>
		<groupId>mysql</groupId>
		<artifactId>mysql-connector-java</artifactId>
		<version>8.0.19</version>
	</dependency>
	<dependency>
		<groupId>org.projectlombok</groupId>
		<artifactId>lombok</artifactId>
		<version>1.18.16</version>
		<scope>provided</scope>
	</dependency>
	<dependency>
		<groupId>log4j</groupId>
		<artifactId>log4j</artifactId>
		<version>1.2.15</version>
		<exclusions>
			<exclusion>
				<groupId>com.sun.jmx</groupId>
				<artifactId>jmxri</artifactId>
			</exclusion>
			<exclusion>
				<groupId>com.sun.jdmk</groupId>
				<artifactId>jmxtools</artifactId>
			</exclusion>
```

```xml
                        <exclusion>
                                <groupId>javax.jms</groupId>
                                <artifactId>jms</artifactId>
                        </exclusion>
                </exclusions>
        </dependency>
        <dependency>
                <groupId>junit</groupId>
                <artifactId>junit</artifactId>
                <version>4.11</version>
                <scope>test</scope>
        </dependency>
</dependencies>

<build>
        <finalName>userLoginRegisterPhaseTwo</finalName>
        <pluginManagement>
                <plugins>
                        <plugin>
                                <artifactId>maven-clean-plugin</artifactId>
                                <version>3.1.0</version>
                        </plugin>
                        <plugin>
                                <artifactId>maven-resources-plugin</artifactId>
                                <version>3.0.2</version>
                        </plugin>
                        <plugin>
                                <artifactId>maven-surefire-plugin</artifactId>
                                <version>2.22.1</version>
                        </plugin>
                        <plugin>
                                <artifactId>maven-install-plugin</artifactId>
                                <version>2.5.2</version>
                        </plugin>
                        <plugin>
                                <artifactId>maven-deploy-plugin</artifactId>
                                <version>2.8.2</version>
                        </plugin>
                        <plugin>
                                <groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-compiler-plugin</artifactId>
                                <configuration>
                                        <source>1.8</source>
                                        <target>1.8</target>
                                </configuration>
                        </plugin>
                        <plugin>
                                <groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-war-plugin</artifactId>
                                <version>3.3.1</version>
                                <configuration>
                                        <warName>user</warName>

<warSourceDirectory>src/main/webapp</warSourceDirectory>
                                </configuration>
                        </plugin>
                  </plugins>
            </pluginManagement>
      </build>
</project>
```

Java Concepts Involved in the project:

1. Classes
2. Exception Handling
3. Packages
4. Logger – log4j
5. Hibernate
6. Servlets
7. JSP
8. Dao-Model-WebServlet-hibernate-JSP

Conclusion:

Login and Registration Application does not crash for incorrect inputs or any discrepancies caused by the user. It has better performance as it uses hibernate, servlets, jsp and logger to log data.