**User Authentication Using Hibernate**

This is an application that lets users authenticate into a web site. It mainly consists of the following

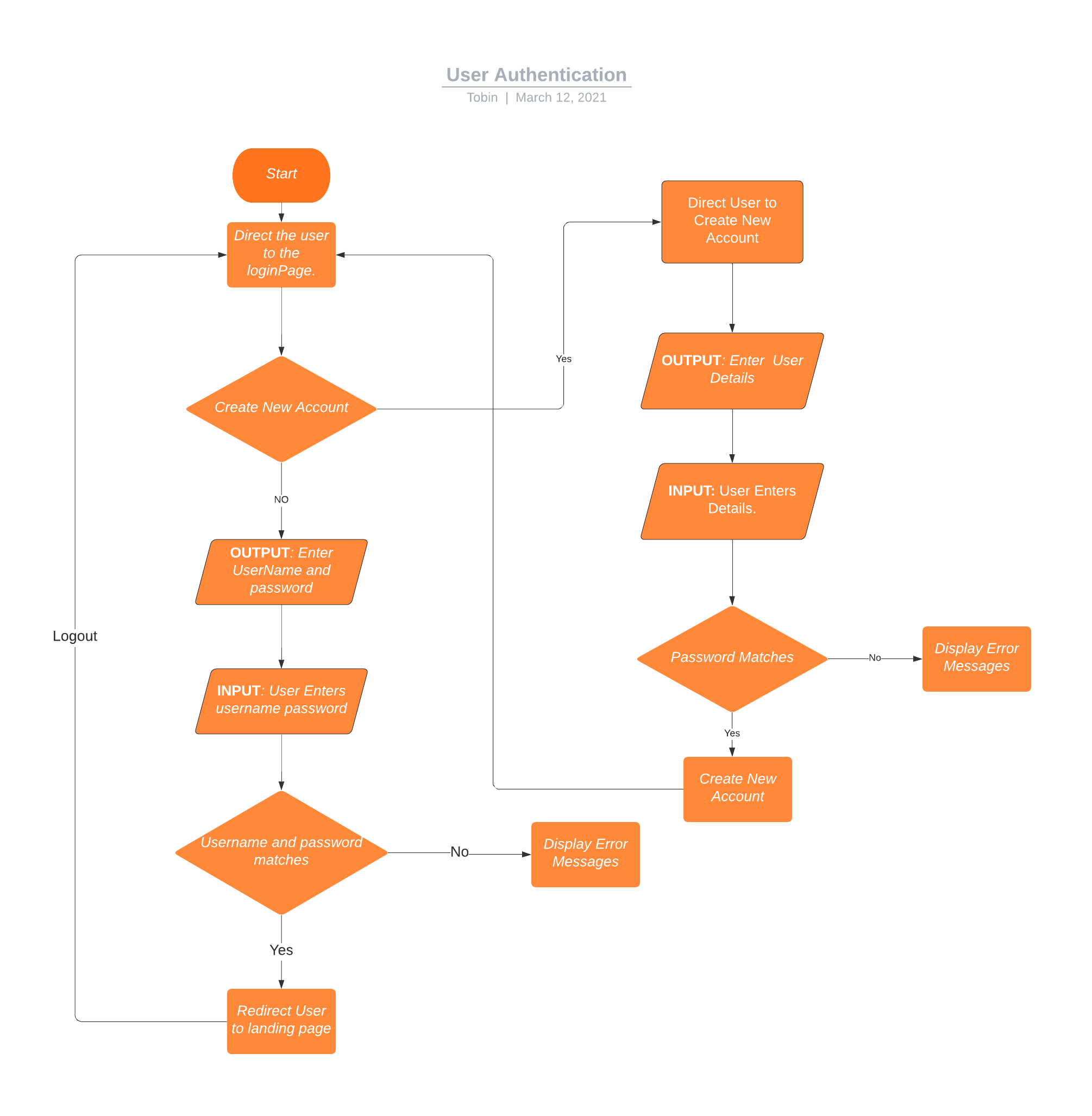
**OVERALL DESCRIPTION**

Login Page: It is the initial page loaded of the website and prompts the user to enter the username and password. The entered inputs are sent to the servlet for verification and upon verification the user is logged in to the application. If the username or password is incorrect the user would not be allowed to log-in and appropriate error messages will be shown to the user. There is also a link in the login page to redirect the user to create an account if the user has none.

Create Account Page: This page can be accessed from the login page via a link. This page consists of fields for the user to enter the username, password and type in the password again for verification. If both the entered passwords match and there is a valid username then the account is created for the user. Appropriate error messages are shown if there is an issue in validating the data entered by the user.

User Landing Page: This page is the landing page on successful authentication of the user. This page also has a link called logout, that redirects the user to Login Page.

**Flow Chart**



**SPRINTS**

SPRINT BACKLOG

|  |  |  |
| --- | --- | --- |
| Day | Tasks | Assignee |
| Not Assigned | Designing Of User Registration HTML page | Sprint Backlog |
| Not Assigned | Coding and testing of Login page | Sprint Backlog |
| Not Assigned | Designing of User Login Page HTML | Sprint Backlog |
| Not Assigned | Coding and Testing of the Login Page | Sprint Backlog |
| Not Assigned | Design of Landing Page HTML | Sprint Backlog |
| Not Assigned | Coding and Designing Of The Landing Page | Sprint Backlog |
| Not Assigned | Integration Of the various Modules and Testing | Sprint Backlog |

**WEEK 1**

After discussing with the Business Analysts and Team leads, we have decided to take the following tasks in week 1 sprint.

|  |  |  |
| --- | --- | --- |
| Day | Tasks | Assignee |
| Day 1-2 | Designing Of User Registration HTML page | Tobin |
| Day 3-4 | Coding and testing of Login page | Tobin |
| Day 5-6 | Designing of User Login Page HTML | Tobin |
| Day 7-8 | Coding and Testing of the Login Page | Tobin |

**After week 1 Status**

|  |  |  |  |
| --- | --- | --- | --- |
| Day | Tasks | Assignee | Status |
| Day 1-2 | Designing Of User Registration HTML page | Tobin | Completed |
| Day 3-4 | Coding and testing of Login page | Tobin | Completed |
| Day 5-6 | Designing of User Login Page HTML | Tobin | Completed |
| Day 7-8 | Coding and Testing of the Login Page | Tobin | Not Completed |

On day 9 and 10 in Sprint 1 the user task were reviewed and checked if they met Acceptance Criteria. The last user story did not meet the requirements hence was moved to next sprint.

On day 10 the retrospective of the sprint was held and various issues and concerns were discussed. It was found that the last task couldn’t be completed to an issue in the UI design provided.

**SPRINT 2**

The following task were decided to be taken in sprint 2.

|  |  |  |
| --- | --- | --- |
| Day | Tasks | Assignee |
| Day 11 - 12 | Coding and Testing of the Login Page (Moved from sprint 1) | Tobin |
| Day 13 – 14 | Design of Landing Page HTML | Tobin |
| Day 15 – 16 | Coding and Designing Of The Landing Page | Tobin |
| Day 17 - 18 | Integration Of the various Modules and Testing | Tobin |

After Sprint 2 development, The status is as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Day | Tasks | Assignee | Status |
| Day 11 - 12 | Coding and Testing of the Login Page (Moved from sprint 1) | Tobin | Completed |
| Day 13 – 14 | Design of Landing Page HTML | Tobin | Completed |
| Day 15 – 16 | Coding and Designing Of The Landing Page | Tobin | Completed |
| Day 17 - 18 | Integration Of the various Modules and Testing | Tobin | Completed |

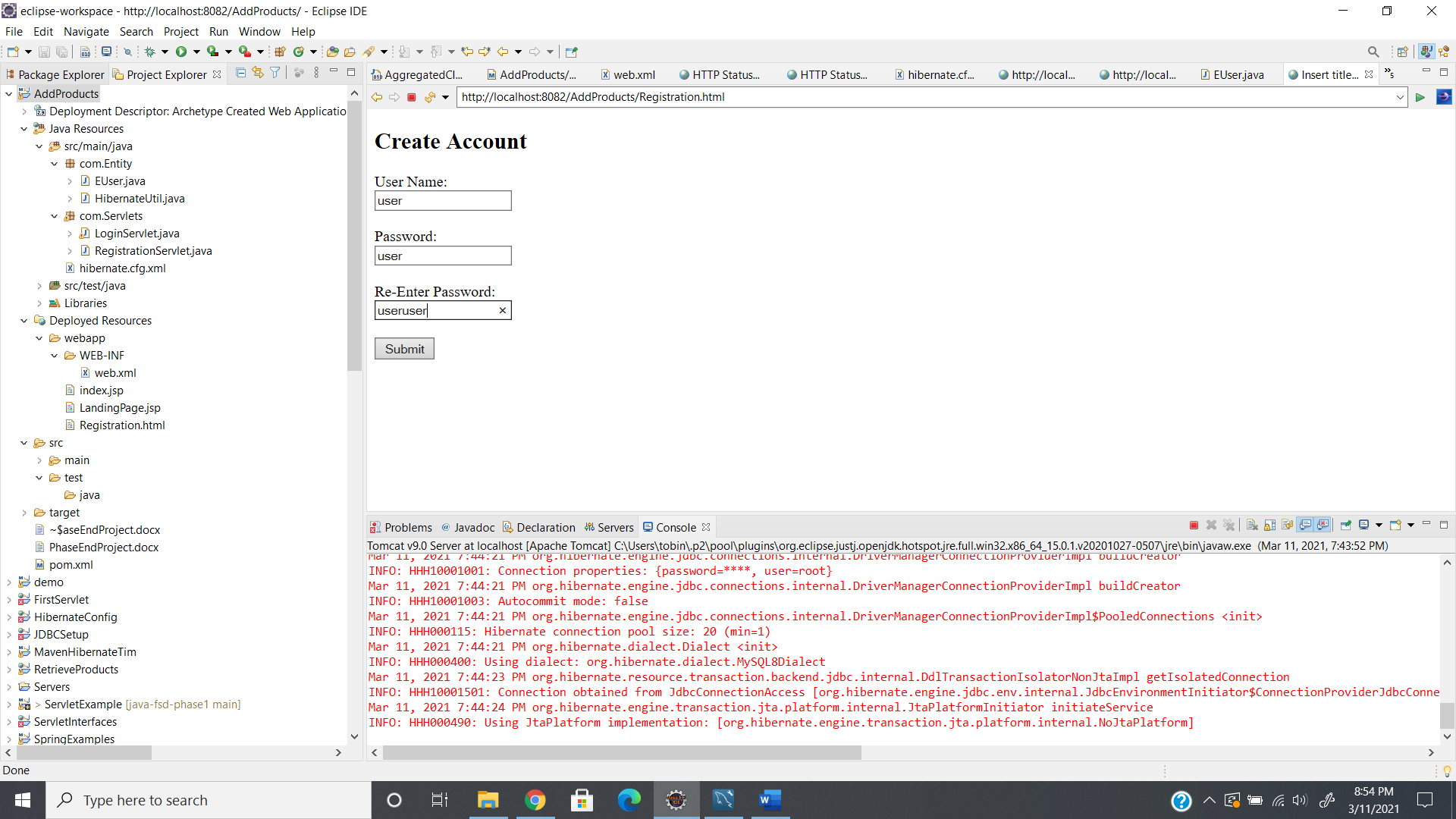
Thus after 2 intense sprints of agile methodology the application was developed successfully.

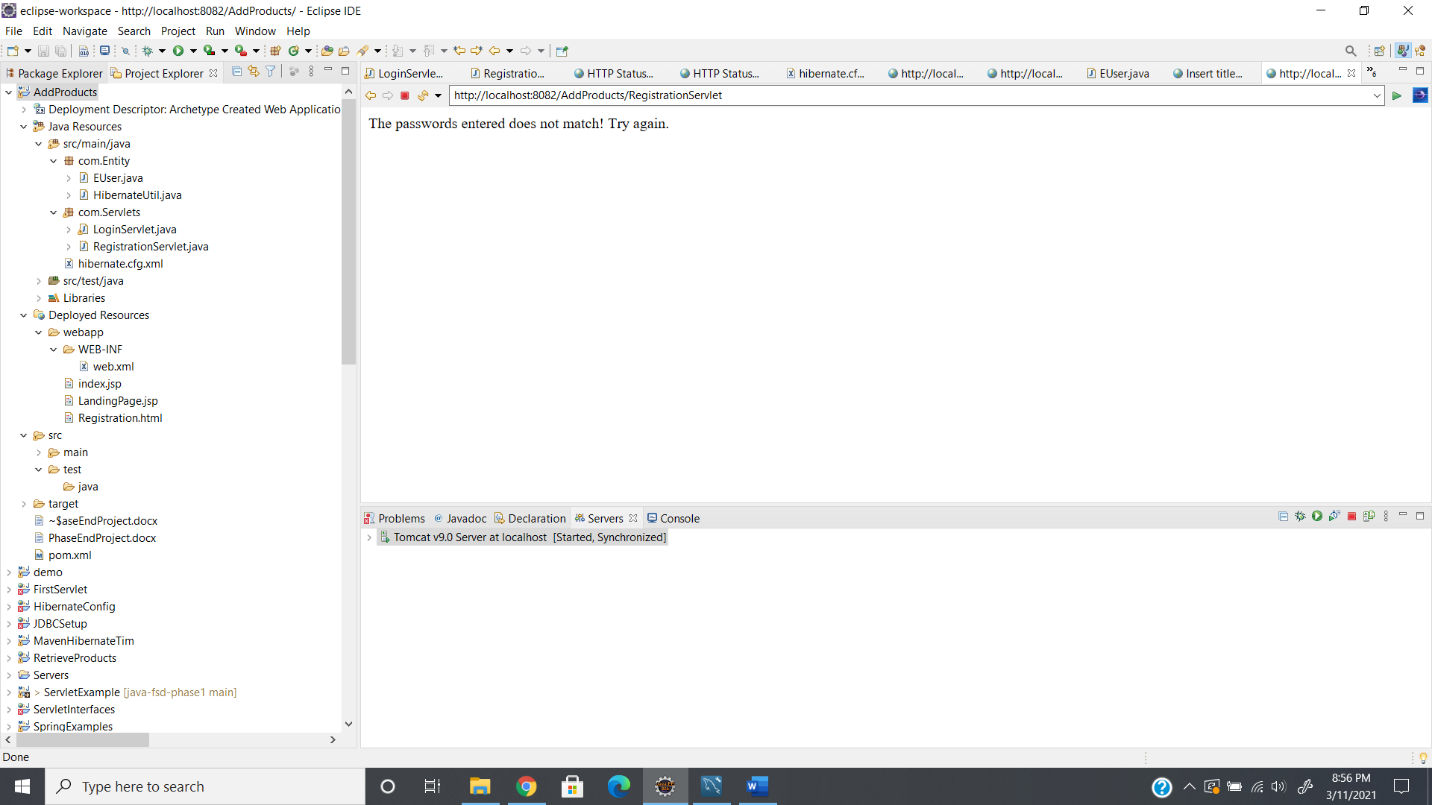
**Conclusion**

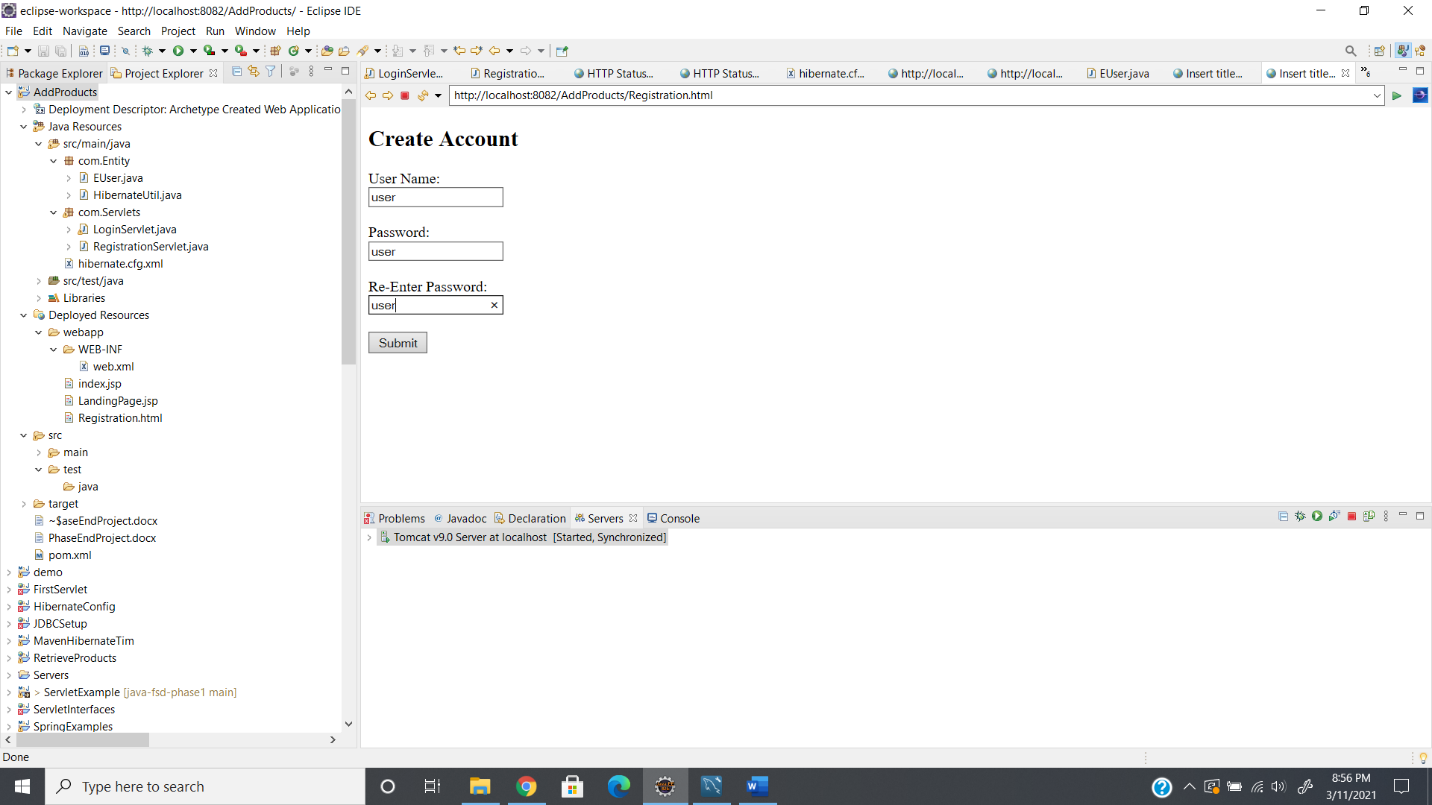
The User authentication website is a an application that let users create accounts and verify accounts.

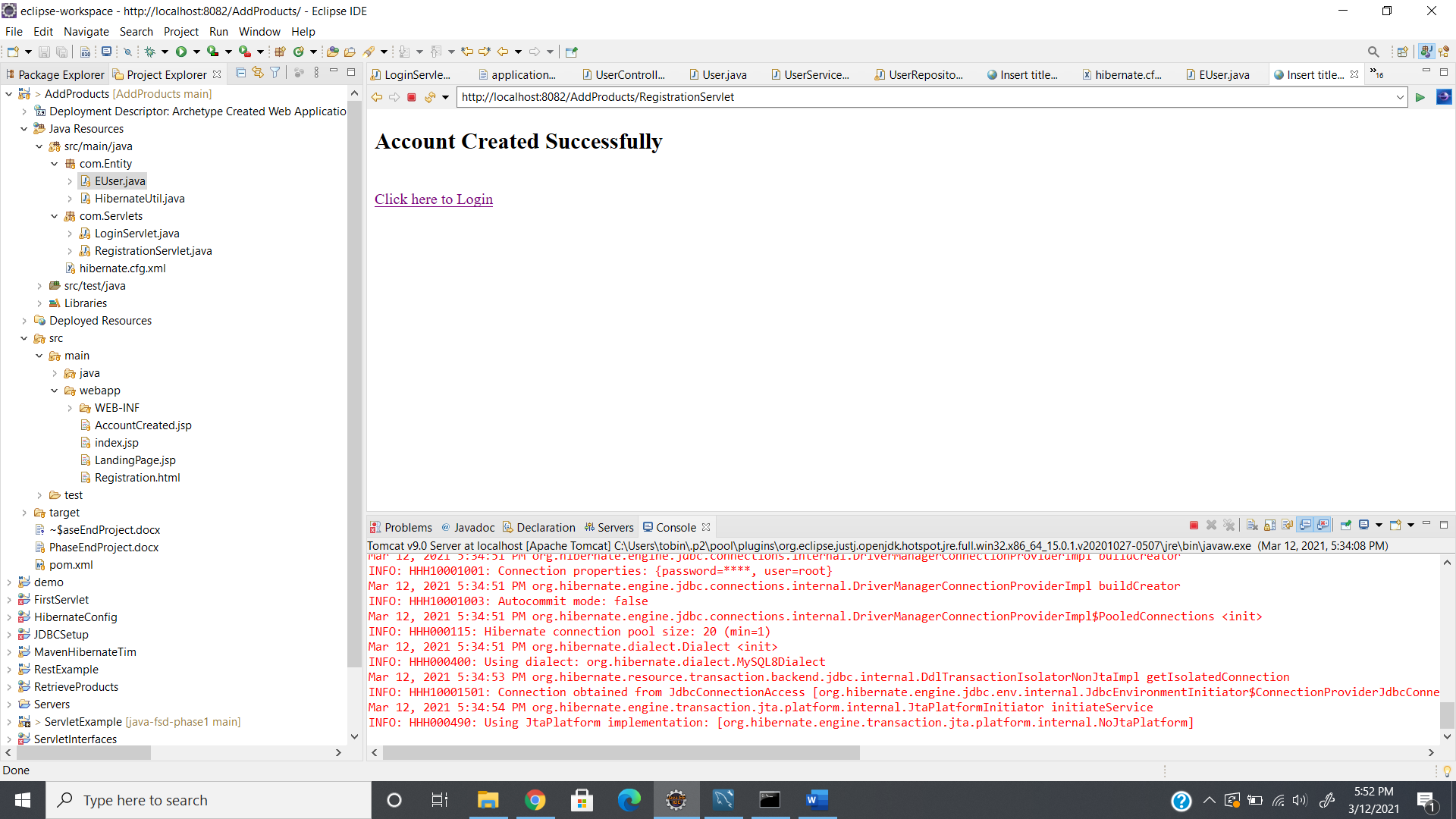
* It is implemented using Java, Hibernate, Servlets, MySql.
* The User table will be created automatically upon running the application by hibernate.
* The Database credentials need to be specified in the configuration file hibernate.cfg.xml.
* The application can be viewed by running on Tomcat Server.

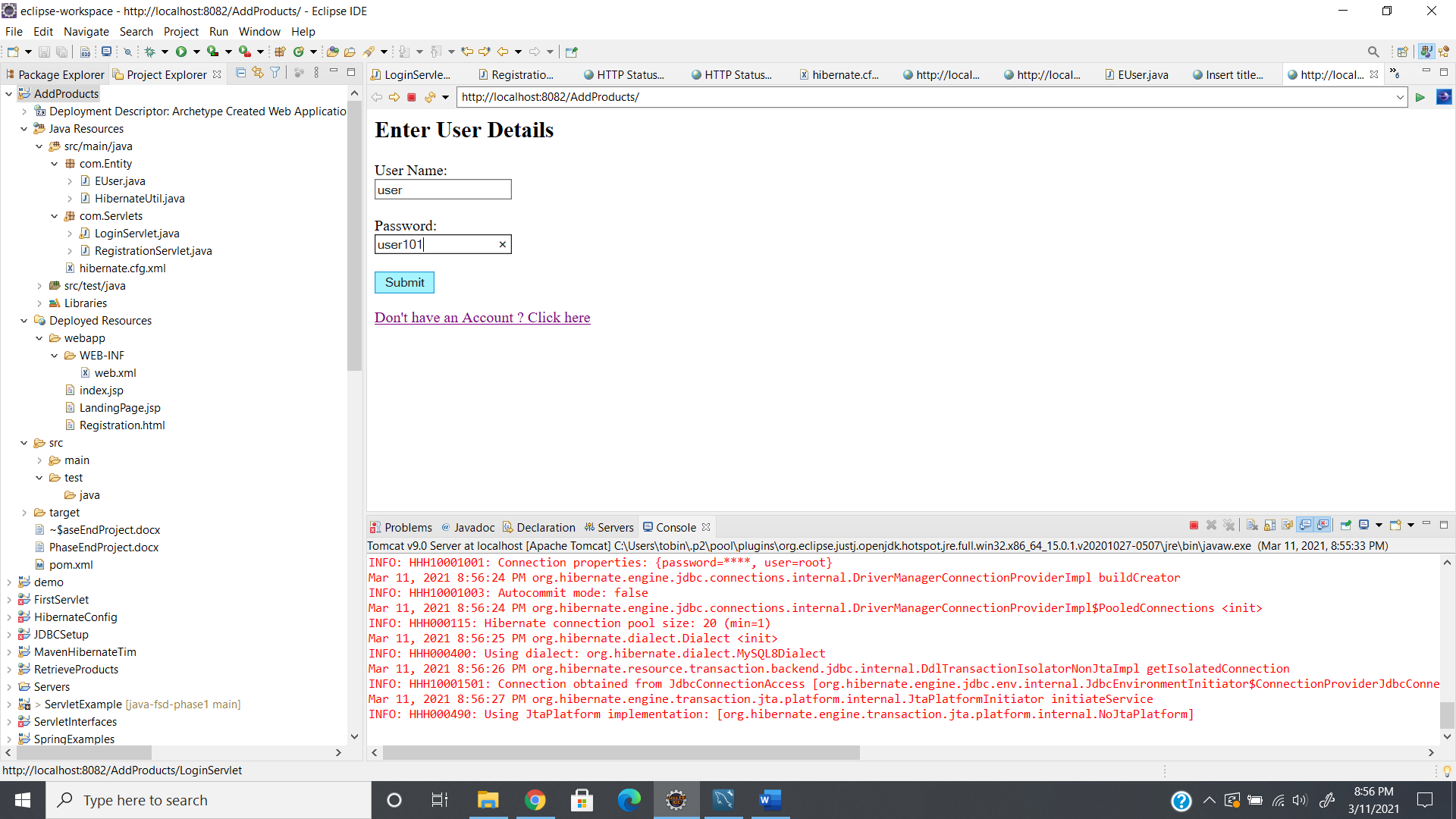
**Screenshots**

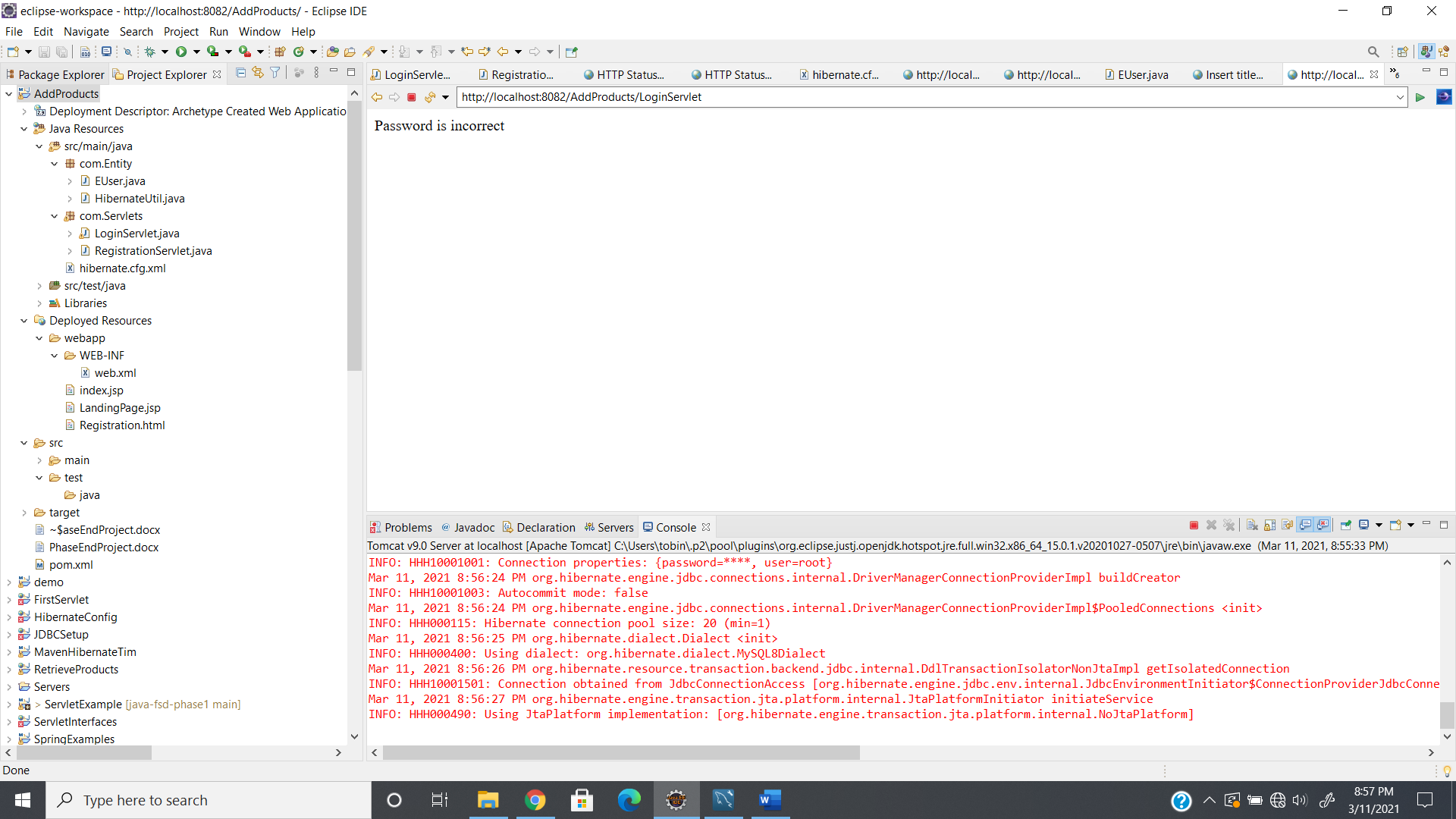
****

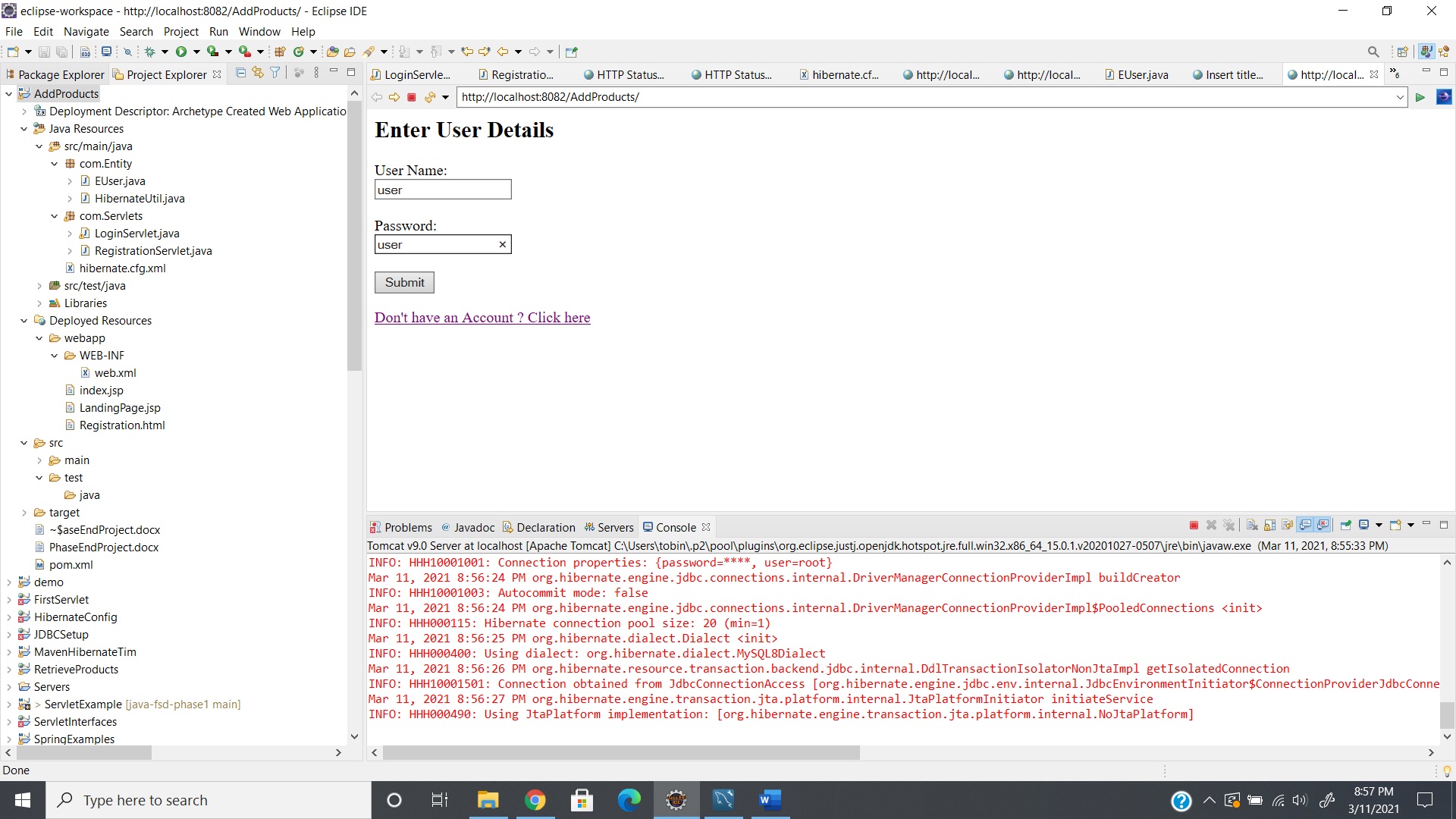
****

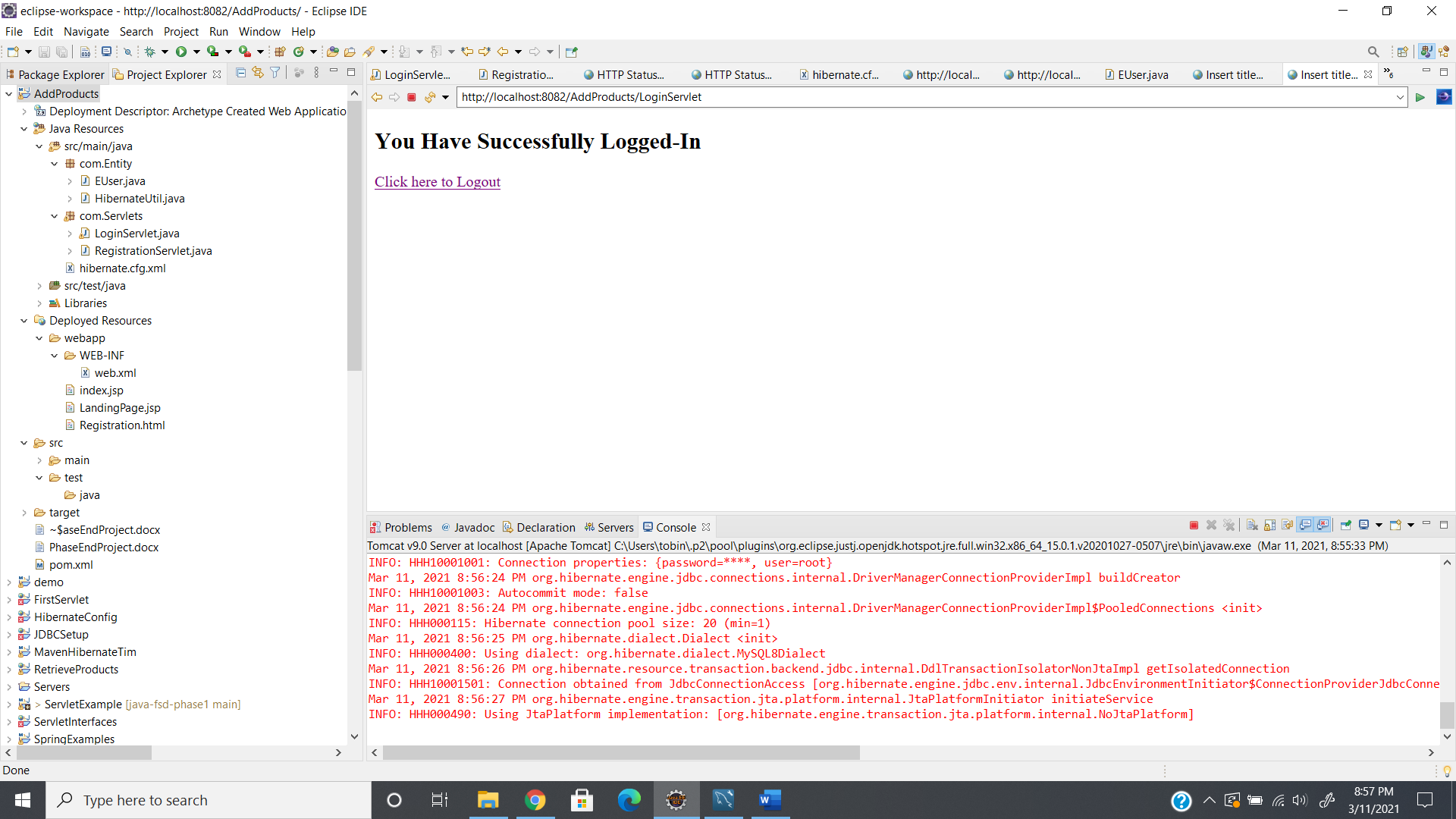
****

****

****

****

****

****

**Source Code:**

EUser.java

package com.Entity;

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 = "user")

public class EUser {

@Id @Column(name="Id")

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int Id;

@Column(name="UserName")

private String UserName;

@Column(name="Password")

private String Password;

public EUser() {

}

public EUser(String userName, String password) {

UserName = userName;

Password = password;

}

public int getId() {

return Id;

}

public void setId(int id) {

Id = id;

}

public String getUserName() {

return UserName;

}

public void setUserName(String userName) {

UserName = userName;

}

public String getPassword() {

return Password;

}

public void setPassword(String password) {

Password = password;

}

}

**LoginServlet**

package com.Servlets;

import java.io.IOException;

import java.util.List;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.query.Query;

import com.Entity.EUser;

import com.Entity.HibernateUtil;

/\*\*

\* Servlet implementation class LoginServlet

\*/

public class LoginServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public LoginServlet() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

response.getWriter().append("Served at: ").append(request.getContextPath());

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

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

// TODO Auto-generated method stub

try {

if(request.getParameter("uname")!="" && request.getParameter("password")!="") {

String userName = request.getParameter("uname");

String password = request.getParameter("password");

SessionFactory factory = HibernateUtil.getSessionFactory();

Session session = factory.openSession();

// using HQL

String hql = "from EUser where UserName = :user\_Name";

Query query = session.createQuery(hql);

query.setParameter("user\_Name", userName);

EUser user = (EUser) query.list().stream().findFirst().orElse(null);

if(user != null) {

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

//response.sendRedirect("LandingPage.jsp");

request.getRequestDispatcher("LandingPage.jsp").forward(request, response);

}

else {

response.getWriter().append("Password is incorrect");

}

}

else {

response.getWriter().append("Invalid User Name");

}

}

else if(request.getParameter("uname")=="") {

response.getWriter().append("Enter User Name");

}

else if(request.getParameter("password")=="") {

response.getWriter().append("Enter Password");

}

}catch(Exception ex) {

}

}

}

**Registration Servlet**

package com.Servlets;

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.query.Query;

import com.Entity.EUser;

import com.Entity.HibernateUtil;

/\*\*

\* Servlet implementation class RegistrationServlet

\*/

public class RegistrationServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public RegistrationServlet() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

response.getWriter().append("Served at: ").append(request.getContextPath());

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

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

// TODO Auto-generated method stub

try {

if(request.getParameter("uname")!="" && request.getParameter("password")!="" && request.getParameter("passwordre")!="") {

if(request.getParameter("password").equals(request.getParameter("passwordre"))) {

String UserName = request.getParameter("uname");

String Password = request.getParameter("password");

EUser eUser = new EUser(UserName, Password);

SessionFactory factory = HibernateUtil.getSessionFactory();

Session session = factory.openSession();

// using HQL

String hql = "from EUser where UserName = :user\_Name";

Query query = session.createQuery(hql);

query.setParameter("user\_Name", UserName);

EUser user = (EUser) query.list().stream().findFirst().orElse(null);

if(user == null) {

session.save(eUser);

request.getRequestDispatcher("AccountCreated.jsp").forward(request, response);

}

else {

response.getWriter().append("An account with the same username already exists!");

}

session.close();

}

else {

response.getWriter().append("The passwords entered does not match! Try again.");

}

}

else if(request.getParameter("uname")=="") {

response.getWriter().append("Please Enter User Name");

}

else if(request.getParameter("password")=="") {

response.getWriter().append("Please Enter Password");

}

else if(request.getParameter("passwordre")=="") {

response.getWriter().append("Please Re-Enter Password");

}

} catch(NumberFormatException nfe)

{

response.getWriter().append("Enter Valid Input");

}

catch(Exception ex) {

response.getWriter().append(ex.getLocalizedMessage());

}

}

}

**GitHub Link:**

https://github.com/tobinsajiantony/Java\_fsd\_Phase\_2\_UserAuthentication