

Table of Contents

Introduction.....	2
Development Environment and Programming Tools	2
Development Environment	2
Programming Language	2
Markup languages	2
Framework and Library.....	3
System Requirement	3
Features	4
Screenshots	4
Steps to create dynamic web project in eclipse.....	4
Steps to install hibernate tools eclipse plugin	7
Steps to create hibernate configuration file using hibernate tools eclipse plugin	8
UI (User Interface).....	12
Source code.....	15
Future Scope	19

ETHIO CLICKS

Introduction

This mini project's goal is to create a dynamic and responsive website for a student information management system (SIMS). This web application is a full-fledged logic application that uses latest technology in the market to provide full functionality, efficiency, and highest quality.

Development Environment and Programming Tools

Development Environment

The development environment we used to develop the mini system are: -

- **Eclipse IDE 2021-03:** - we prefer this IDE since It is the fastest Java IDE and This IDE will help us to create a dynamic web application.
- **Apache Tomcat 9.0:** - we prefer this server because it is open-source, and it's free to use. Easy to install and configure. Multiple applications can run at the same time without any issues.

Programming Language

- **Java Programming Language**
We will use Java Server Page (JSP) and Servlet to implement our web application.
- **Hibernate Query Language (HQL)**
Hibernate Query Language (HQL) is an object-oriented query language, similar to SQL, but instead of operating on tables and columns, HQL works with persistent objects and their properties. HQL queries are translated by Hibernate into conventional SQL queries, which in turns perform action on database.

Markup languages

- **HTML and CSS**
we will use HTML (Hypertext markup language) to build the structure of our website and CSS to style our website.

Framework and Library

- Bootstrap
- mysql-connector-java-8.0.25
- antlr-2.7.7
- byte-buddy-1.10.22
- classmate-1.5.1
- FastInfoset-1.2.15
- hibernate-commons-annotations-5.1.2.Final
- hibernate-core-5.5.0.Final
- istack-commons-runtime-3.0.7
- jandex-2.2.3.Final
- javassist-3.27.0-GA
- javax.activation-api-1.2.0
- javax.persistence-api-2.2
- jaxb-api-2.3.1
- jaxb-runtime-2.3.1
- jboss-logging-3.4.2.Final
- jboss-transaction-api_1.2_spec-1.1.1.Final
- stax-ex-1.8
- txw2-2.3.1

System Requirement

On client-side

- Operating System (Any)
- Web Browser (IE, Mozilla Firefox, Google Chrome, Safari, Opera ...)

On server-side

- Web Server (Apache Tomcat)
- Hibernate
- MySQL

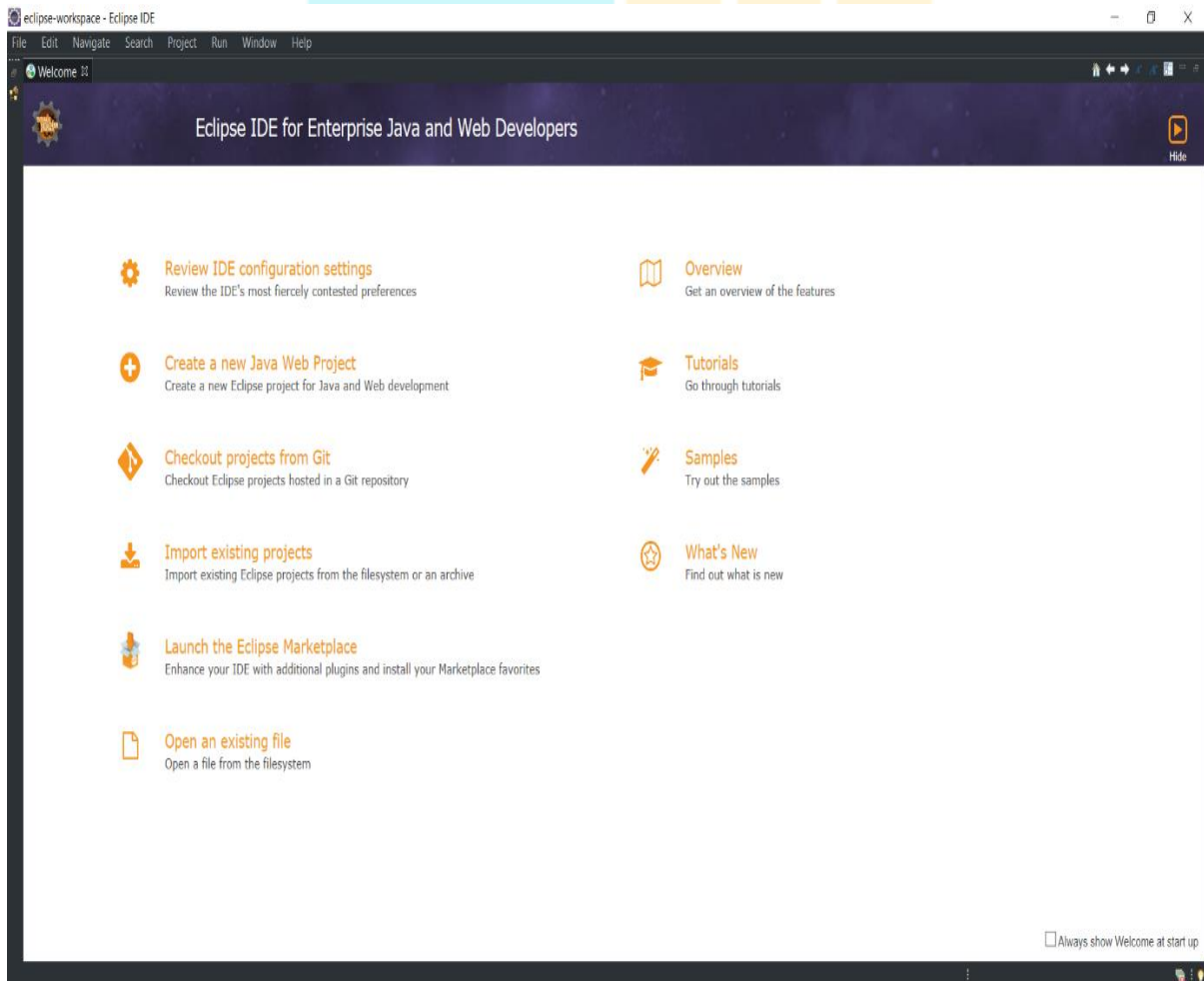
Features

- Login
- Create an Account
- Register Student.
- Display List of Student.
- Search Student
- Logout

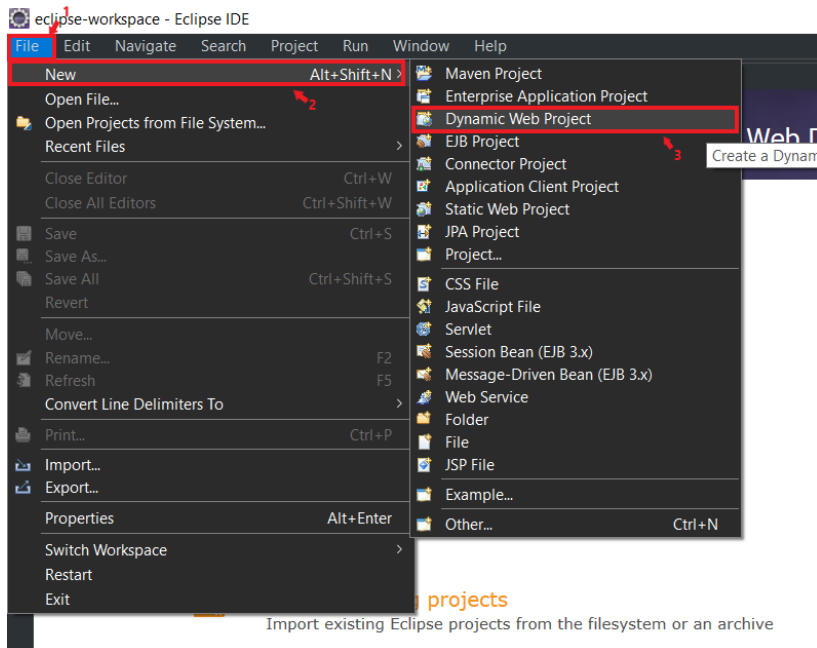
Screenshots

Steps to create dynamic web project in eclipse

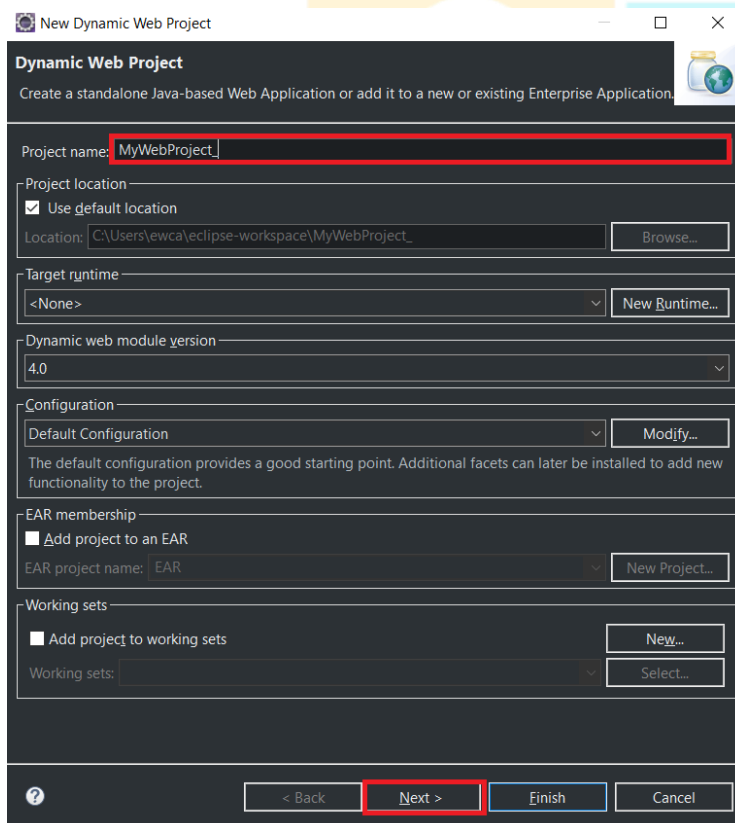
Step 1: Launch Eclipse IDE for Enterprise Java and Web Developers.



Step 2: Open Dynamic Web Project, Click File > New > Dynamic Web Project as shown in the figure.

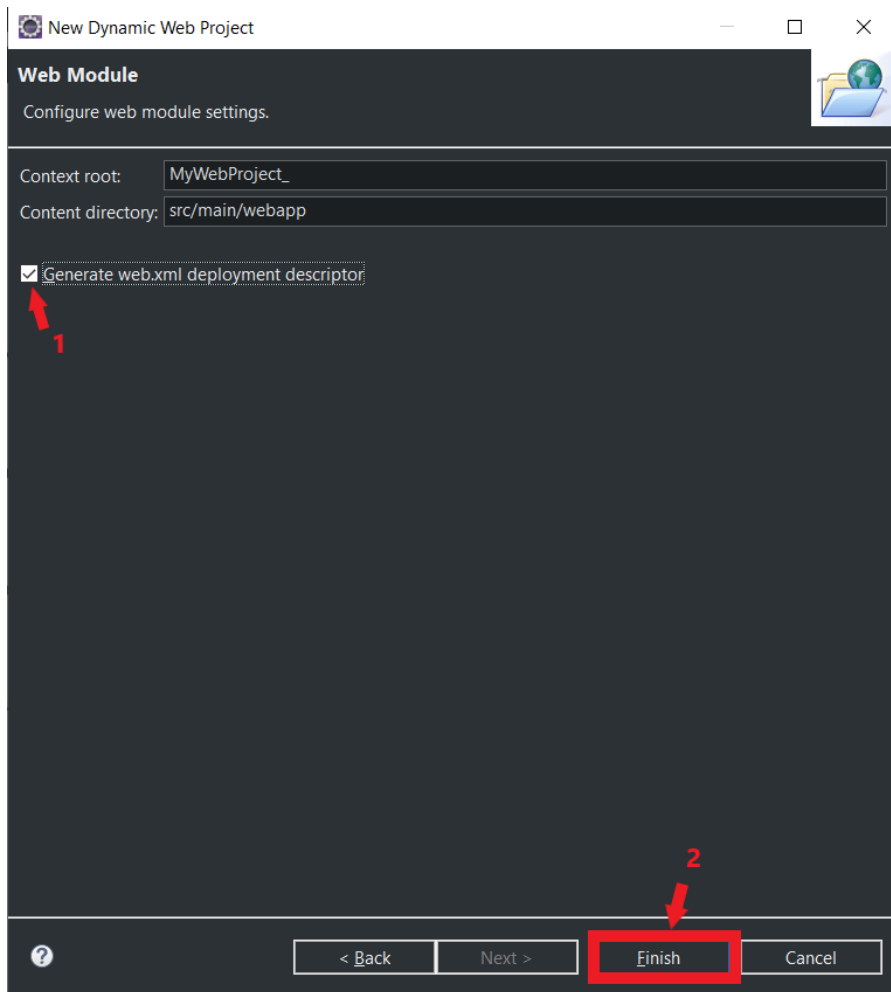


Step: 3 Name the project e.g. MyWebProject.

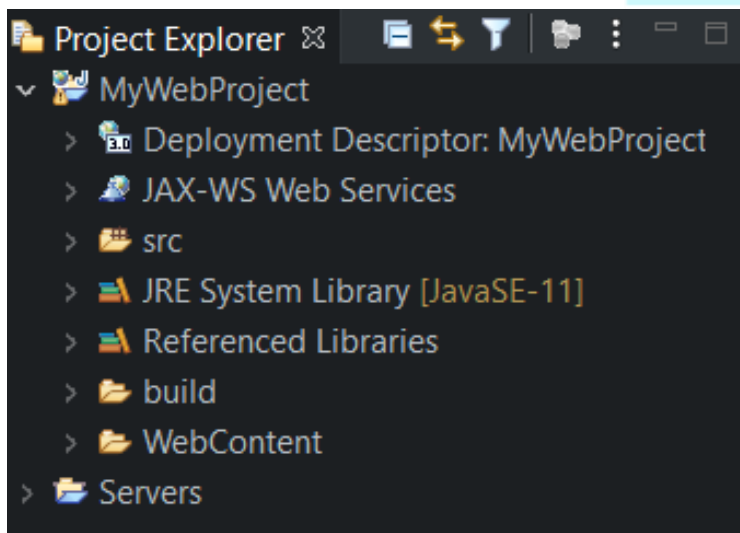


Then click “Next”.

After that select “Generate web.xml deployment descriptor” and click “Finish”.

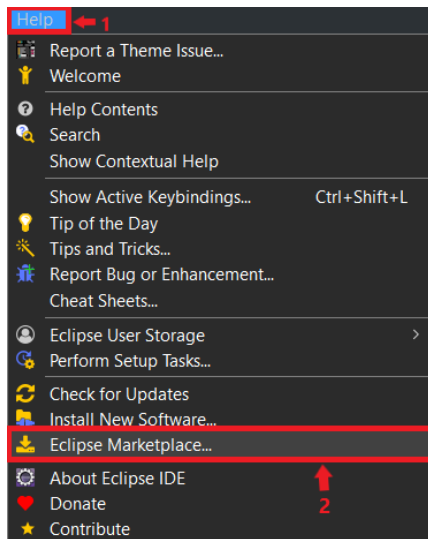


After that you will see something like this in the project explorer tab.



Steps to install hibernate tools eclipse plugin

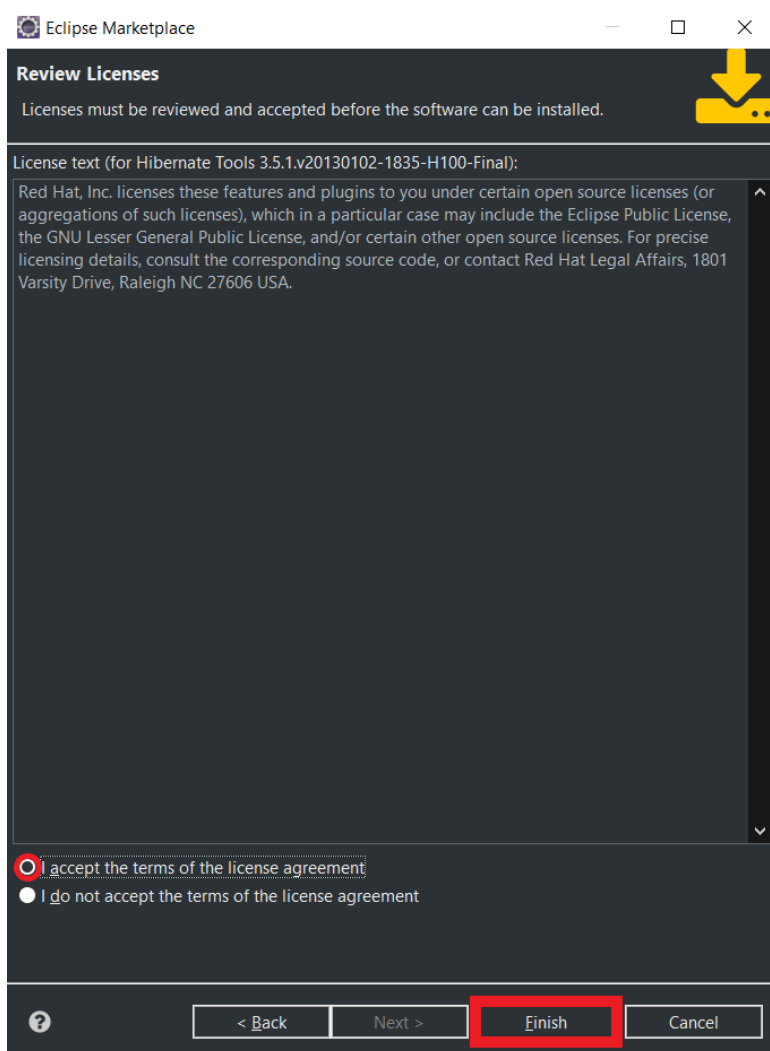
Step 1: Go to “Eclipse Marketplace” from the Help Menu, as shown in below image.



Step 2: Use the search option to find the “Hibernate Tools” plugin, hibernate plugins are Eclipse version specific. So make sure you choose the same one as your eclipse. You can find Eclipse version from “About Eclipse” popup page.



Step 3: Click on the install button, make sure the “I accept the terms of the license agreement” radio box is selected and click on “Finish” button.

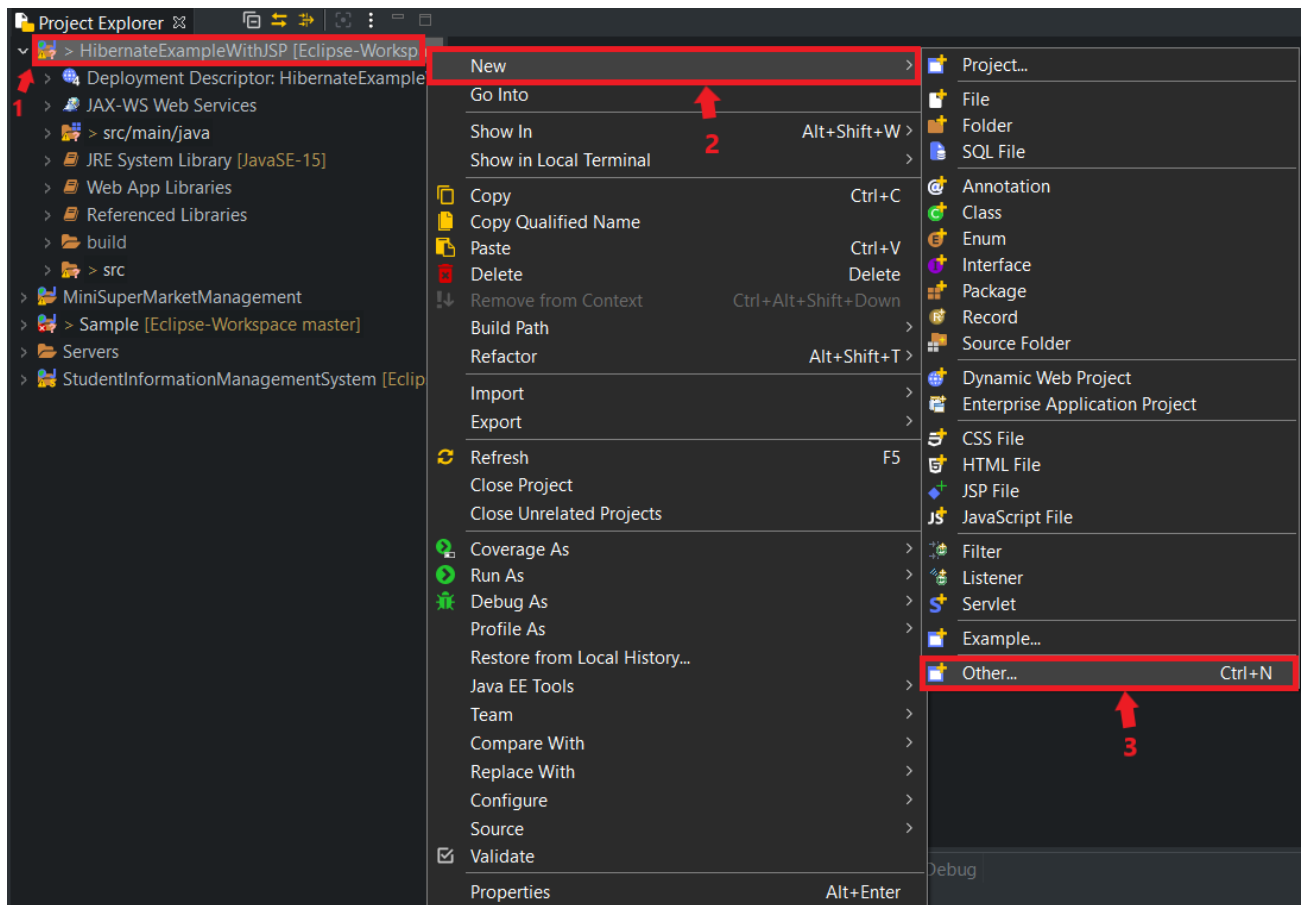


Once the plugin is installed, it will ask to restart the Eclipse. Just restart it and you are ready to use it for hibernate projects.

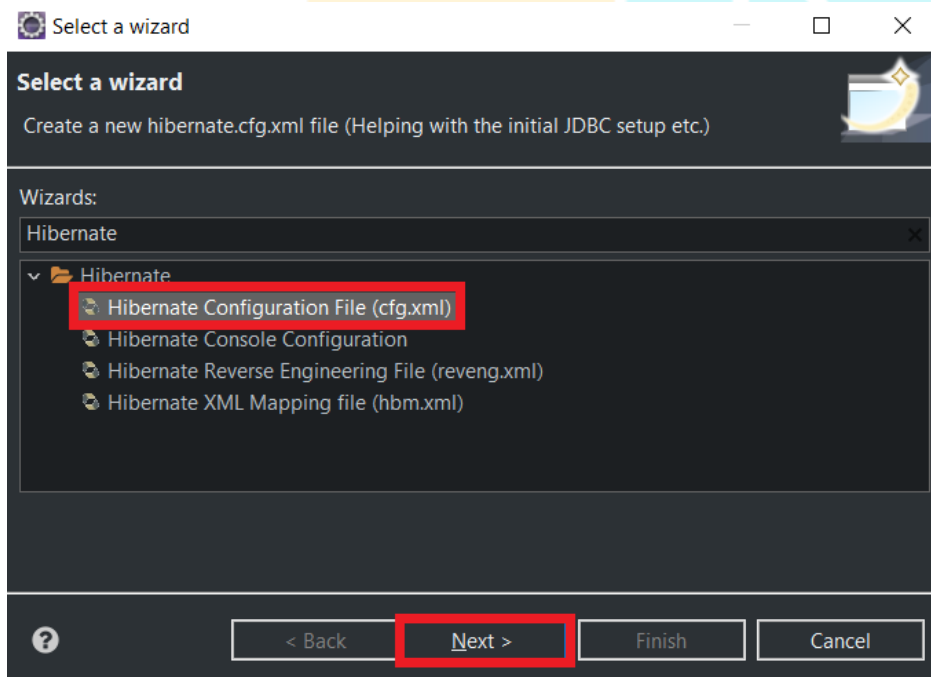
Steps to create hibernate configuration file using hibernate tools eclipse plugin

Now that hibernate tools plugin is installed, let's see how to create hibernate configuration file using it.

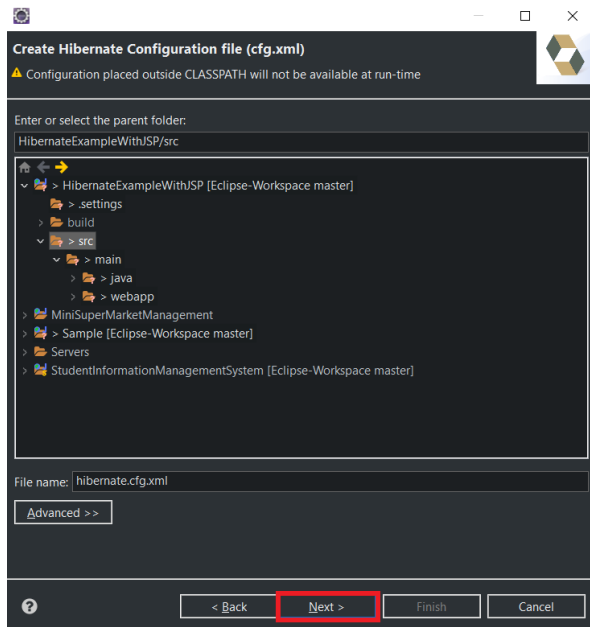
Step 1: Right click on the project and go to “New” > “Other” or you can open this window using keyboard Command+N (Mac), Ctrl+N (Windows).



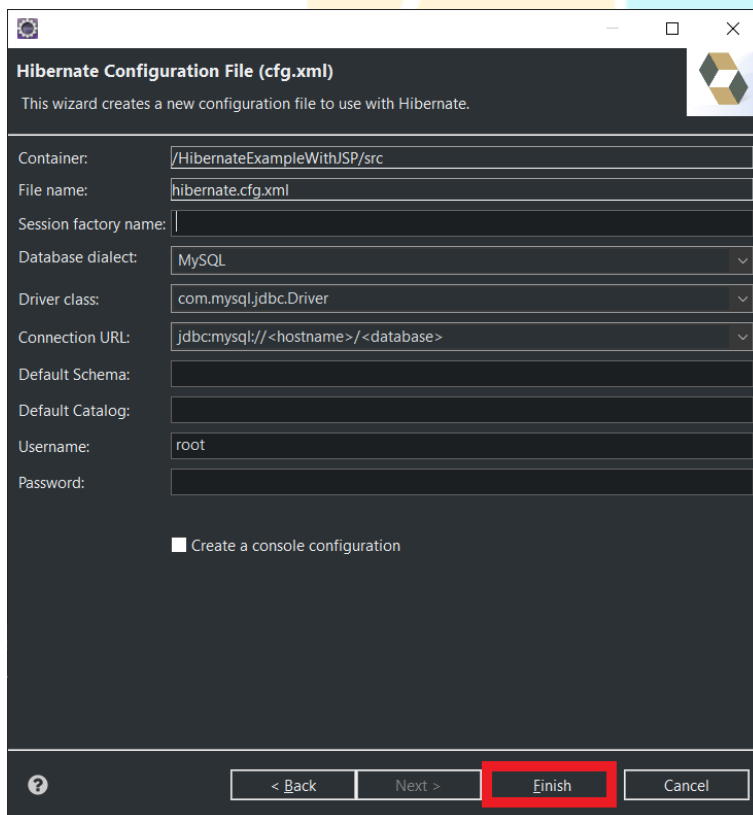
In the popup window, select “Hibernate Configuration File” as shown in below image.



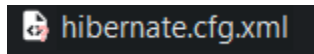
Step 2: In the next window, you can select the parent folder and set the file name, as shown below.



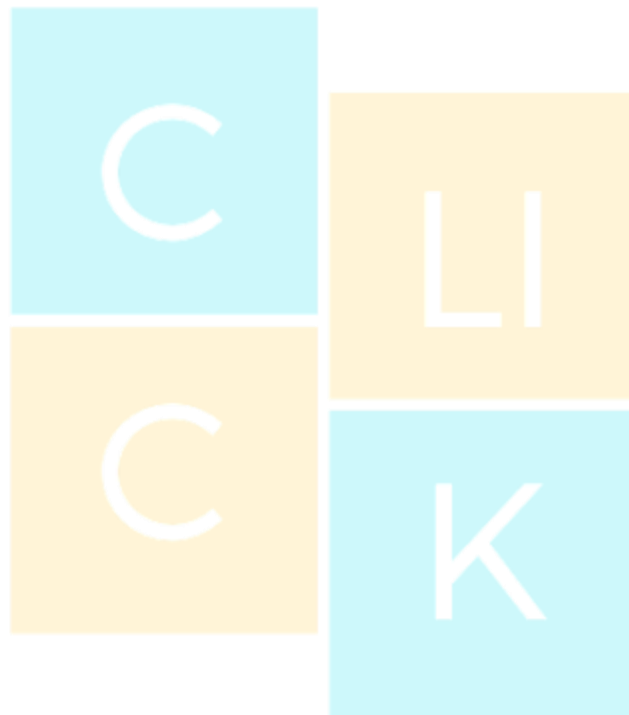
Step 3: The next window lets you set the database properties such as dialect, database user, password and connection URLs.



Step 4: When you click “Finish” button, hibernate.cfg.xml file will get added in the parent folder where your java src is present.

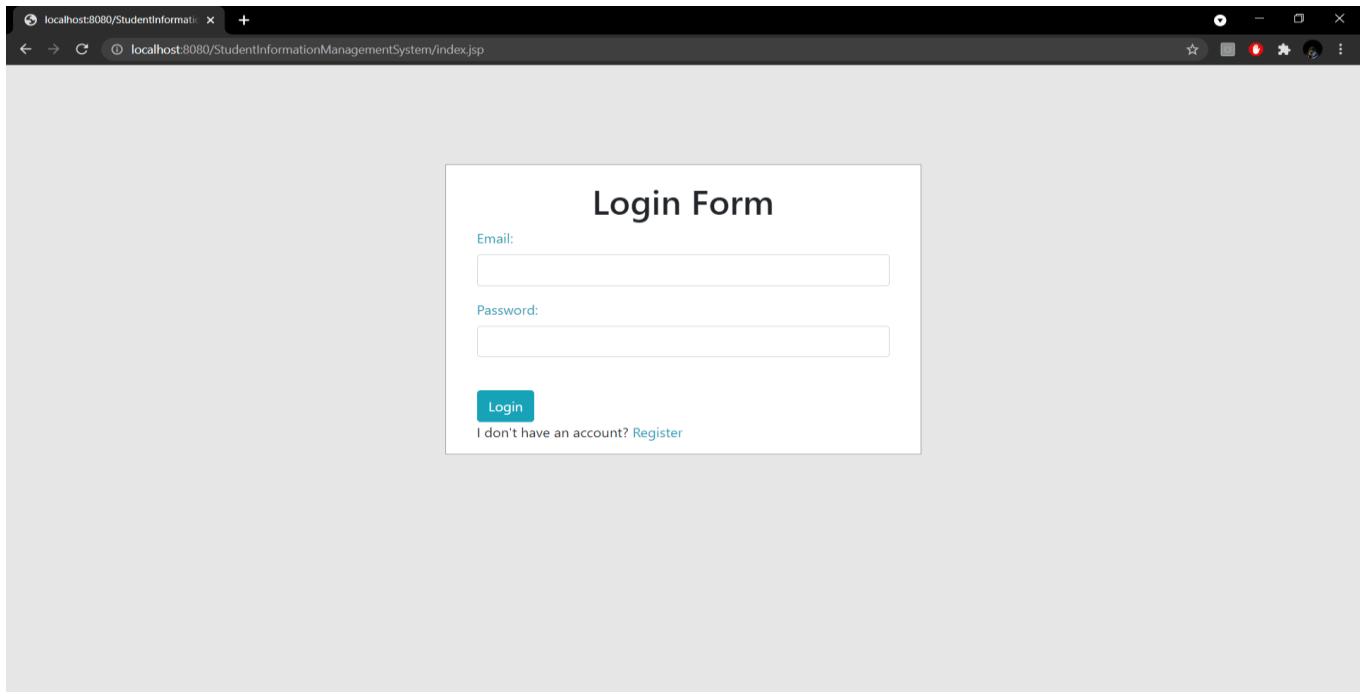


For more tutorial [Click Here](#).



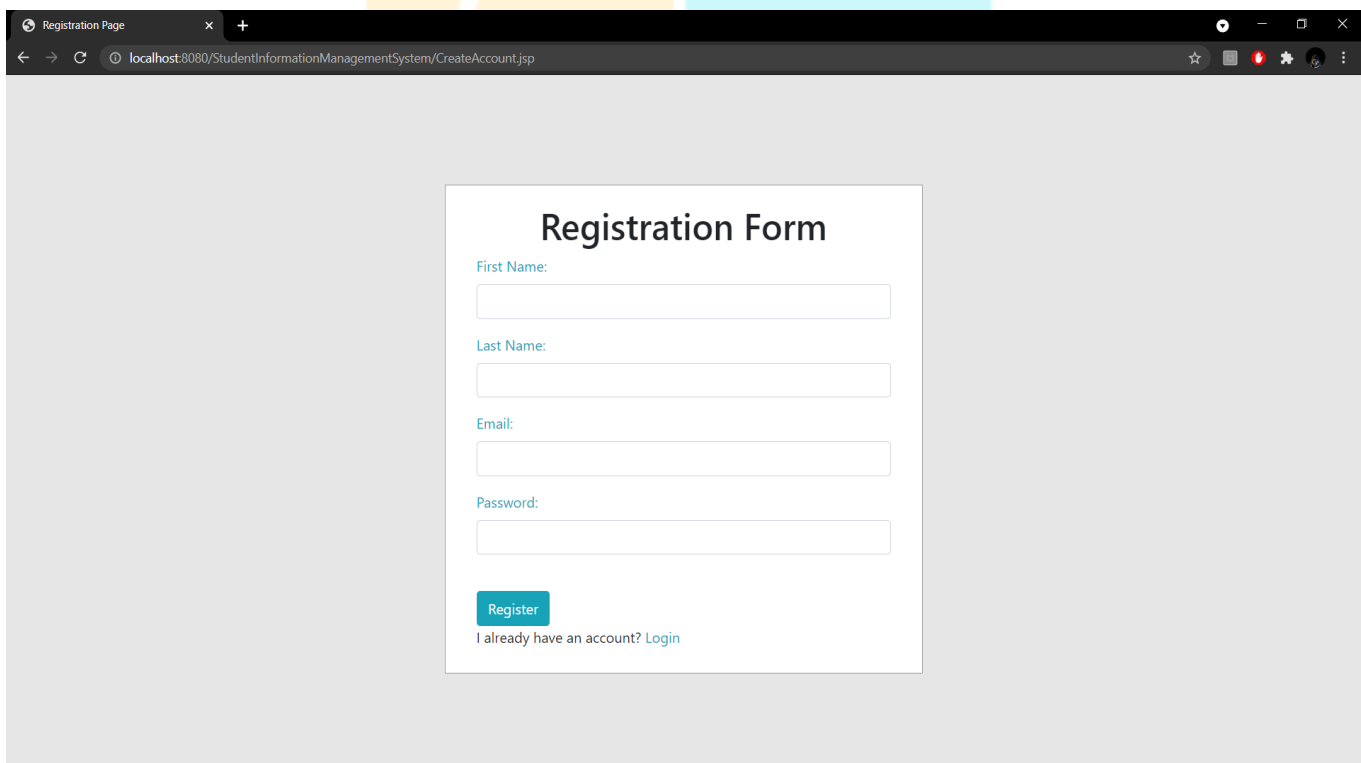
ETHIO CLICKS

UI (User Interface)



A screenshot of a web browser displaying the login page. The browser's address bar shows the URL `localhost:8080/StudentInformationManagementSystem/index.jsp`. The page has a light gray background. In the center, there is a white rectangular box titled "Login Form". Inside this box, there are two input fields: "Email:" and "Password:". Below the "Password:" field is a blue "Login" button. At the bottom of the box, there is a link that says "I don't have an account? Register".

Figure 1: Login Page



A screenshot of a web browser displaying the registration page. The browser's address bar shows the URL `localhost:8080/StudentInformationManagementSystem/CreateAccount.jsp`. The page has a light gray background. In the center, there is a white rectangular box titled "Registration Form". Inside this box, there are four input fields: "First Name:", "Last Name:", "Email:", and "Password:". Below the "Password:" field is a blue "Register" button. At the bottom of the box, there is a link that says "I already have an account? Login".

Figure 2: Registration Page

Invalid username and/or password Please [Try Again!](#)



Figure 3: Error Page

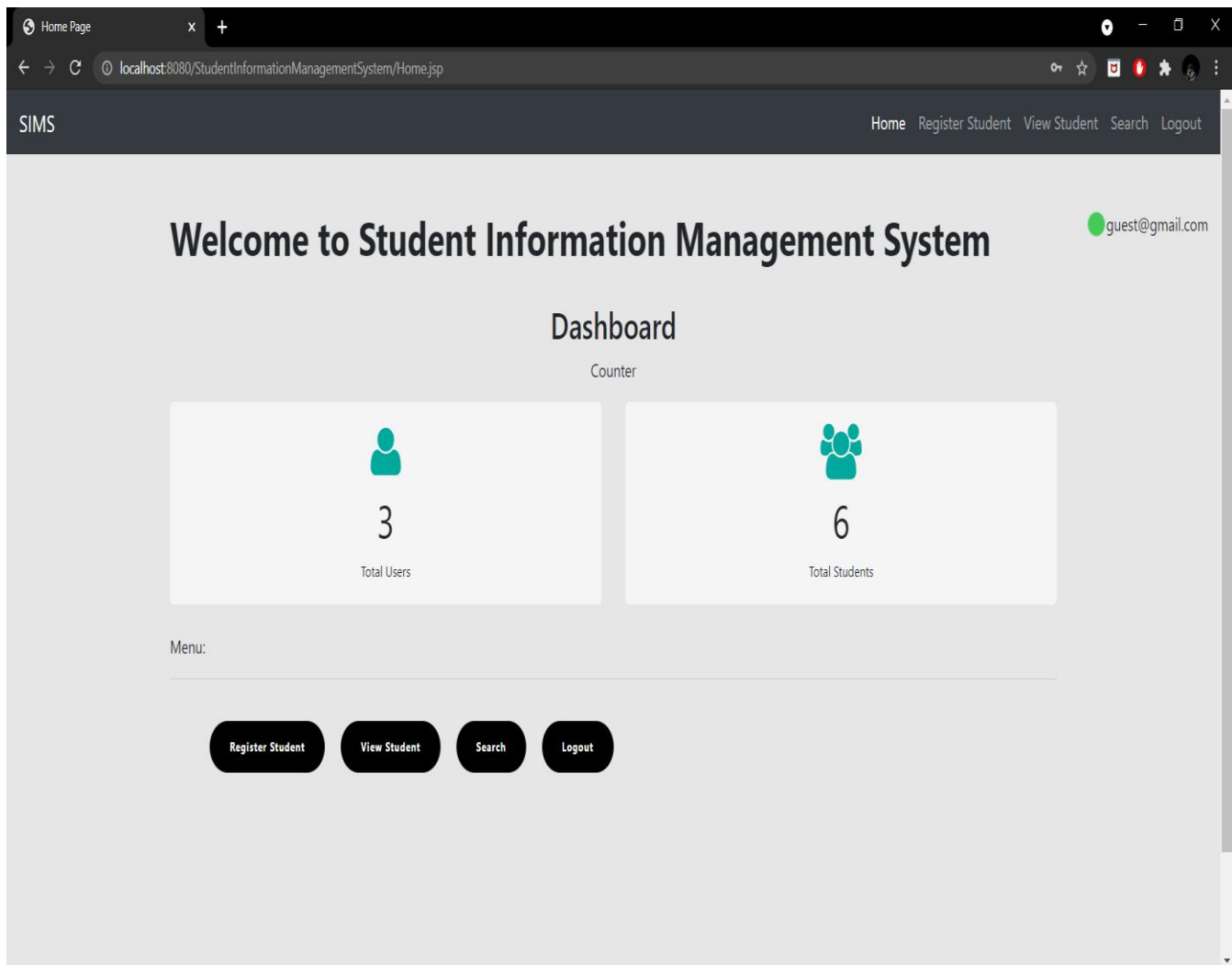
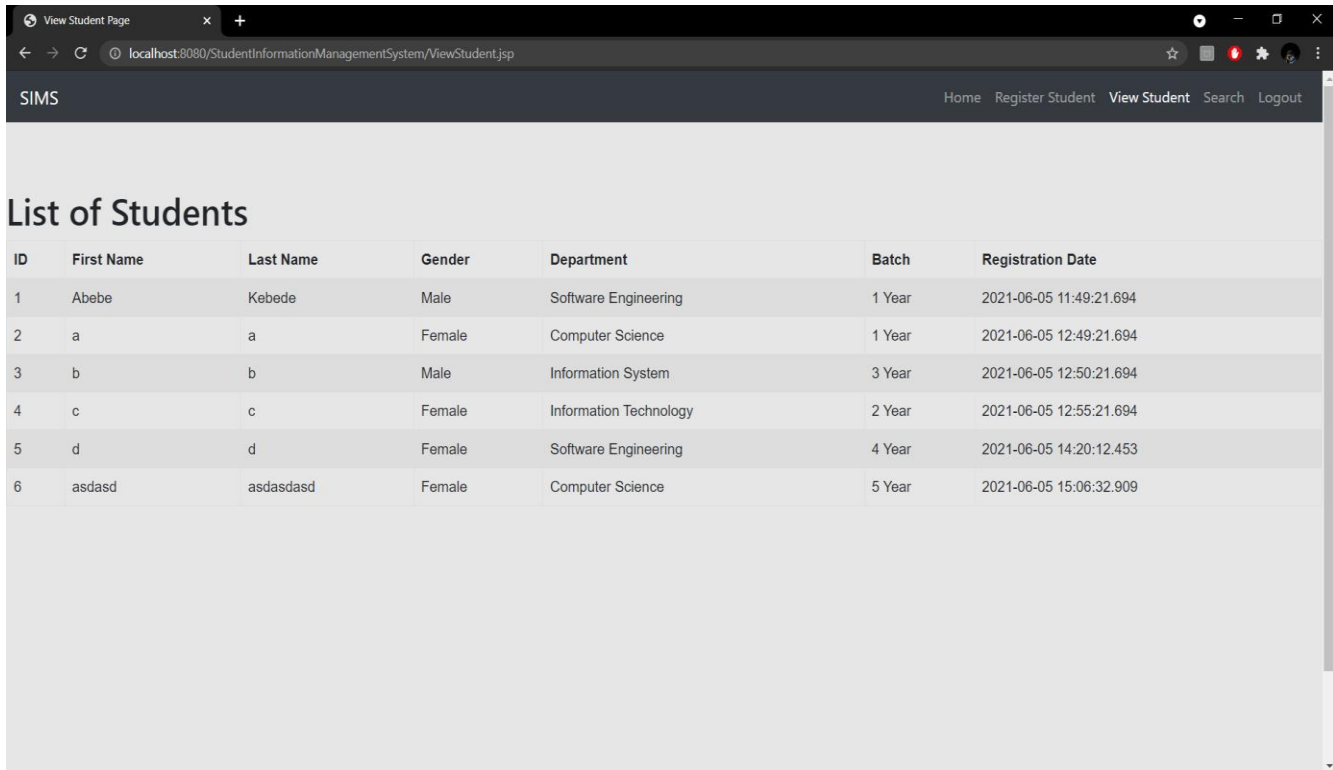


Figure 4: Home Page



View Student Page

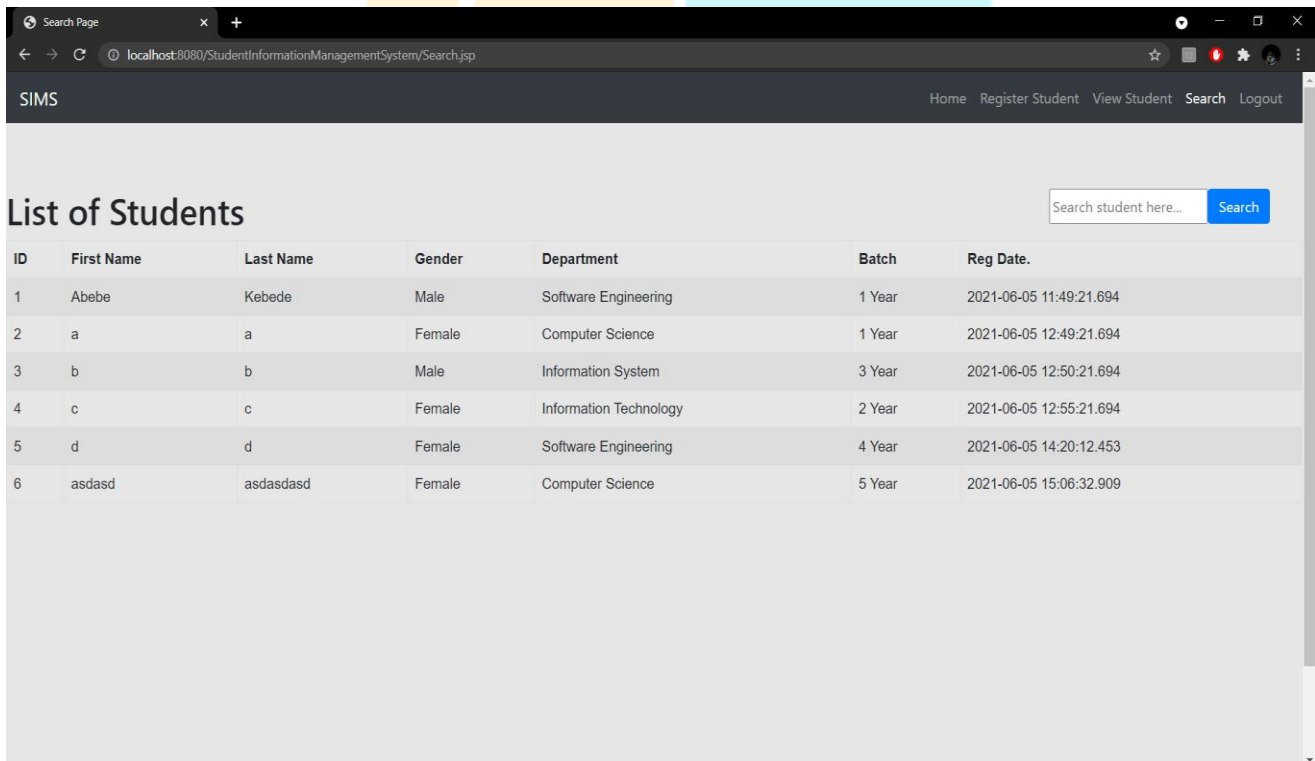
localhost:8080/StudentInformationManagementSystem/ViewStudent.jsp

SIMS Home Register Student View Student Search Logout

List of Students

ID	First Name	Last Name	Gender	Department	Batch	Registration Date
1	Abebe	Kebede	Male	Software Engineering	1 Year	2021-06-05 11:49:21.694
2	a	a	Female	Computer Science	1 Year	2021-06-05 12:49:21.694
3	b	b	Male	Information System	3 Year	2021-06-05 12:50:21.694
4	c	c	Female	Information Technology	2 Year	2021-06-05 12:55:21.694
5	d	d	Female	Software Engineering	4 Year	2021-06-05 14:20:12.453
6	asdasd	asdasdasd	Female	Computer Science	5 Year	2021-06-05 15:06:32.909

Figure 5: Display list of registered student.



Search Page

localhost:8080/StudentInformationManagementSystem/Search.jsp

SIMS Home Register Student View Student Search Logout

List of Students

Search student here... Search

ID	First Name	Last Name	Gender	Department	Batch	Reg Date.
1	Abebe	Kebede	Male	Software Engineering	1 Year	2021-06-05 11:49:21.694
2	a	a	Female	Computer Science	1 Year	2021-06-05 12:49:21.694
3	b	b	Male	Information System	3 Year	2021-06-05 12:50:21.694
4	c	c	Female	Information Technology	2 Year	2021-06-05 12:55:21.694
5	d	d	Female	Software Engineering	4 Year	2021-06-05 14:20:12.453
6	asdasd	asdasdasd	Female	Computer Science	5 Year	2021-06-05 15:06:32.909

Figure 6: Search student page.

Source code

User Validation (LoginHandler.java)

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

    PrintWriter out = response.getWriter();
    String uname = request.getParameter("email");
    String pw = request.getParameter("pass");

    try {
        UserInfo user = new UserInfo();
        Configuration conf = new Configuration();
        conf.configure("hibernate.cfg.xml");
        Session session = conf.buildSessionFactory().openSession();
        Transaction transaction = session.beginTransaction();
        user = UserInfo session.createQuery("FROM UserInfo u WHERE u.email
=:Email and u.password =:Password").setString("Email", uname).setString("Password",
pw).uniqueResult();

        transaction.commit();
        session.close();

        if (user == null){
            response.sendRedirect("LoginError.jsp");
        }
        else{
            request.getSession().setAttribute("uname",uname);
            response.sendRedirect("Home.jsp");
        }
    } catch (Exception e) {
        out.print("Error:" + e.getMessage());
    }
}
```

Create an account (Regiser.java)

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

    PrintWriter out = response.getWriter();
    try {
        out.println("Working");
        UserInfo user = new UserInfo();
        Date date = new Date();
        Timestamp timeStamp = new Timestamp(date.getTime());
        String fname = request.getParameter("fname");
        user.setFirstName(fname);
        String lname = request.getParameter("lname");
        user.setLastName(lname);
        String email = request.getParameter("email");
        user.setEmail(email);
    }
```

```

request.getSession().setAttribute("uname" email);
String pw = request.getParameter("password");
user.setPassword(pw);
//set the value of timeStamp
user.setTimeStamp(timeStamp);

//Hibernate Configuration
Configuration conf = new Configuration();
conf.configure("hibernate.cfg.xml");

Session session = conf.buildSessionFactory().openSession();
Transaction transaction = session.beginTransaction();
//Save to DB
session.save(user);

transaction.commit();
session.close();

out.println("Done");

request.getSession().setAttribute("uname", email);
response.sendRedirect("Home.jsp");
}catch(Exception e){
    out.println("Error: "+e.getMessage());
}
}

```

Add Students into database (RegisterStudentHandler.java)

```

protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    PrintWriter out = response.getWriter();
    try {
        out.println("Working...");
        StudentInfo student = new StudentInfo();
        Date date = new Date();
        Timestamp timeStamp = new Timestamp(date.getTime());
        String fname = request.getParameter("firstname");
        student.setFirstName(fname);
        String lname = request.getParameter("lastname");
        student.setLastName(lname);
        String gender = request.getParameter("gender");
        student.setGender(gender);
        String dept = request.getParameter("department");
        student.setDepartment(dept);
        String batch = request.getParameter("batch");
        student.setBatch(batch);
        String desc = request.getParameter("description");
        student.setDescription(desc);
        //TimeStamp
        student.setTimeStamp(timeStamp);

        // Hibernate configuration
        Configuration conf = new Configuration();
    }
}

```



```

        conf.configure("hibernate.cfg.xml");

        Session session = conf.buildSessionFactory().openSession();
        Transaction transaction = session.beginTransaction();
        //Save to DB
        session.save(student);

        transaction.commit();
        session.close();

        out.println("Done");
        response.sendRedirect("/HibernateExampleWithJSP/ViewStudent");

    } catch (Exception e) {
        out.println("Error: " + e.getMessage());
    }
}

```

Display List of Student (ViewStudent.java)

```

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

    try {
        //Create list of user using ArrayList
        List<StudentInfo> listOfStudent = new ArrayList<>();
        //Hibernate Configuration
        Configuration conf = new Configuration();
        conf.configure("hibernate.cfg.xml");
        //Start session
        Session session = conf.buildSessionFactory().openSession();
        //Begin a unit of work and return the associated Transaction object.
        Transaction transaction = session.beginTransaction();
        //HQL get student from database
        listOfStudent = session.createQuery("FROM StudentInfo").getResultList();
        //commit transaction
        transaction.commit();
        //End the session by releasing the JDBC connection and cleaning up
        session.close();

        System.out.println("Done");

        request.getSession().setAttribute("List", listOfStudent);
        response.sendRedirect("ViewStudent.jsp");

    } catch (Exception e) {
        System.out.println("Error: " + e.getMessage());
    }
}

```

Search Student Page (SearchStudent.java)

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

```

        try {
            //Create list of student using ArrayList
            List<StudentInfo> listOfStudent = new ArrayList<>();
            //Hibernate Configuration
            Configuration conf = new Configuration();
            conf.configure("hibernate.cfg.xml");
            //Start session
            Session session = conf.buildSessionFactory().openSession();
            // Begin a unit of work and return the associated Transaction object.
            Transaction transaction = session.beginTransaction();
            //Get user input from TextField
            String tfSearch = request.getParameter("searchTxt");
            //HQL select all user from database
            String query = "FROM StudentInfo as si";
            listOfStudent = (List<StudentInfo>) session.createQuery(query).getResultList(); ;
            if (tfSearch != null)
                //HQL select all where FirstName = user input
                String query2 = "FROM StudentInfo as si WHERE si.firstName like :searchField";
                listOfStudent = (List<StudentInfo>)
                session.createQuery(query2).setString("searchField", "%" + tfSearch +
                "%").getResultList();

            }
            // commit transaction
            transaction.commit();
            // End the session by releasing the JDBC connection and cleaning up
            session.close();

            System.out.println("Done");

            request.getSession().setAttribute("List", listOfStudent);
            response.sendRedirect("Search.jsp");

        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }

    }
}

```

Logout Page (Logout.jsp)

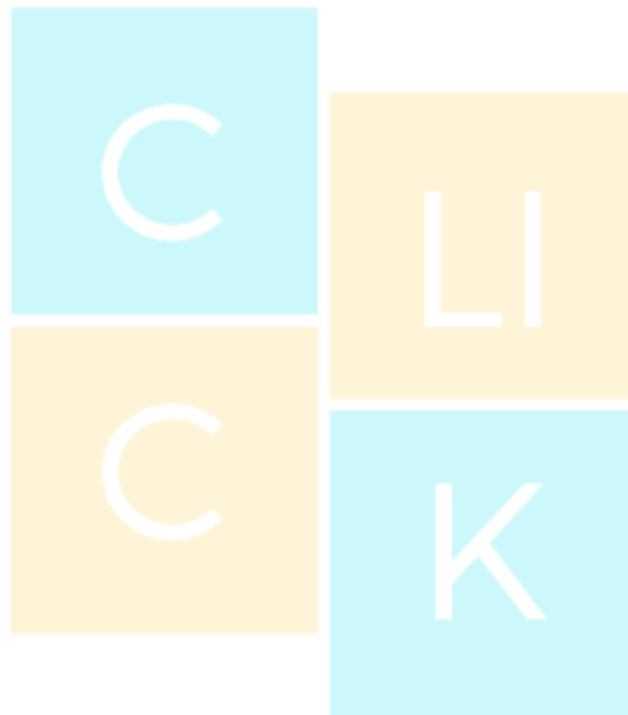
```

<%
    session.removeAttribute("uname");
    session.invalidate();
    response.sendRedirect("index.jsp");
%>

```

Future Scope

- Action (Edit & Delete).
- Improving Security (Password Encryption).
- User validation using email and password recovery.
- Normalization (One to One, One to Many, Many to Many Relation...) .
- Print student information.



ETHIO CLICKS