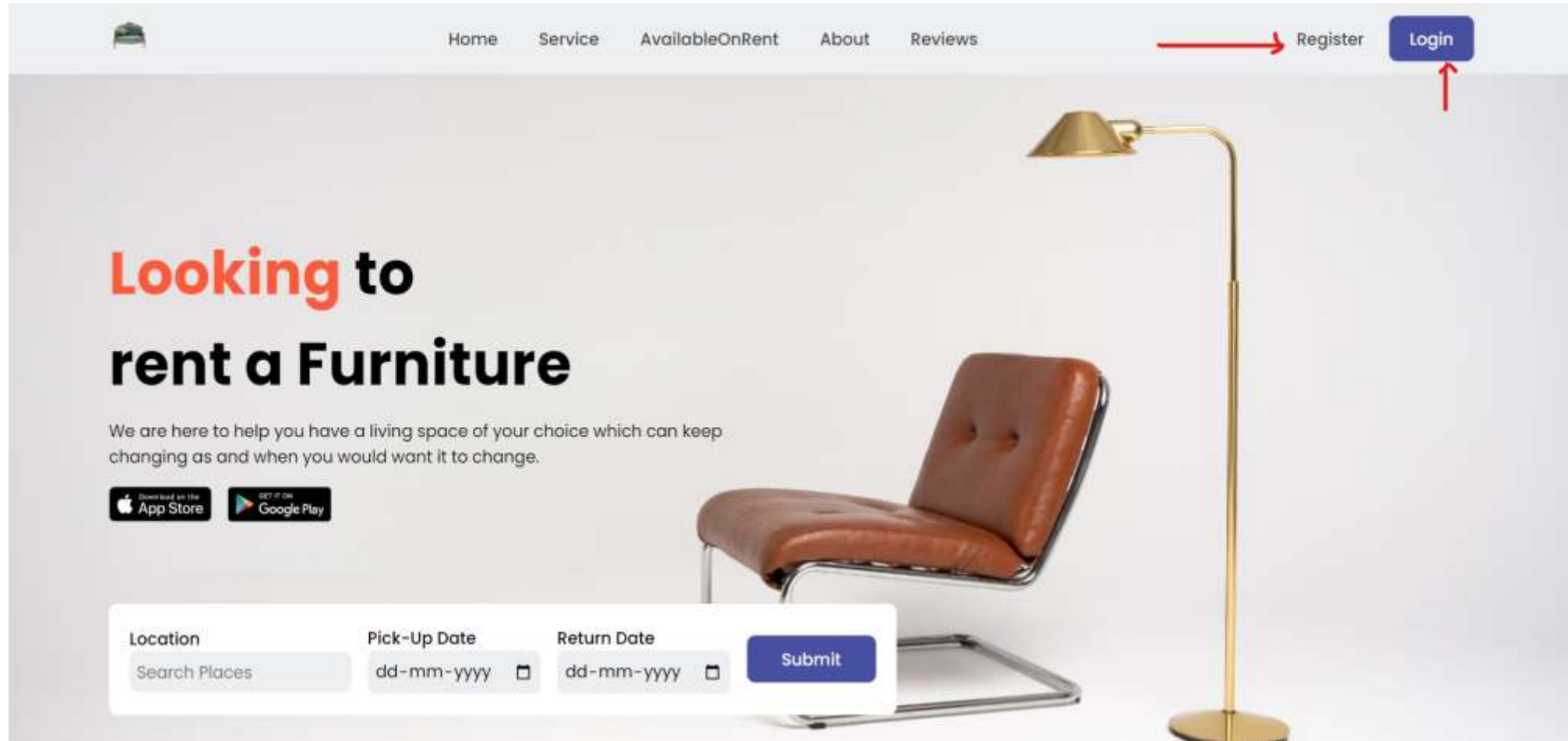


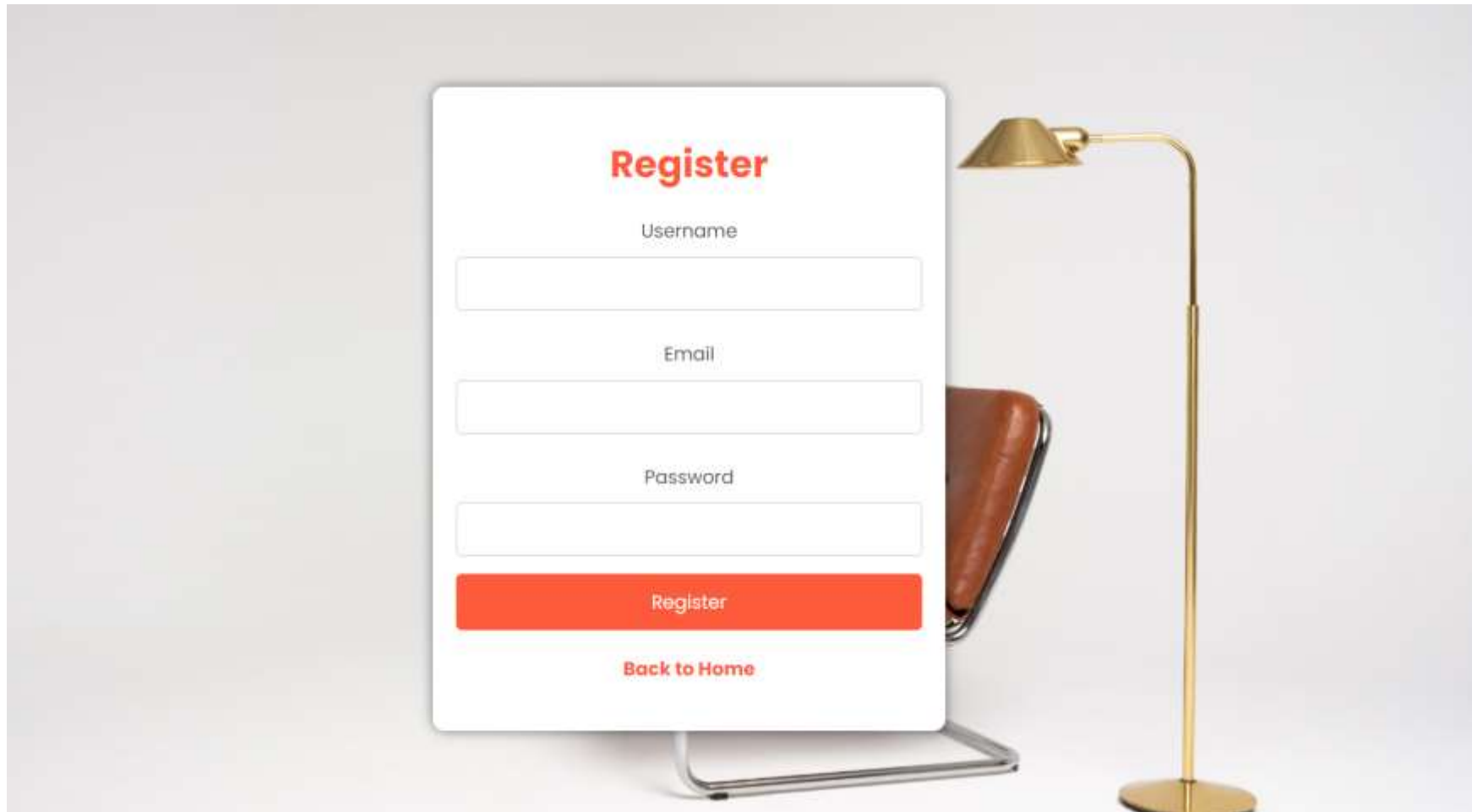
FULL STACK PROJECT

Step:1 Expected out put

This is our web application with the help of two buttons named Register and Login.



This is our register page by giving the name and email id and by creating the password we can register

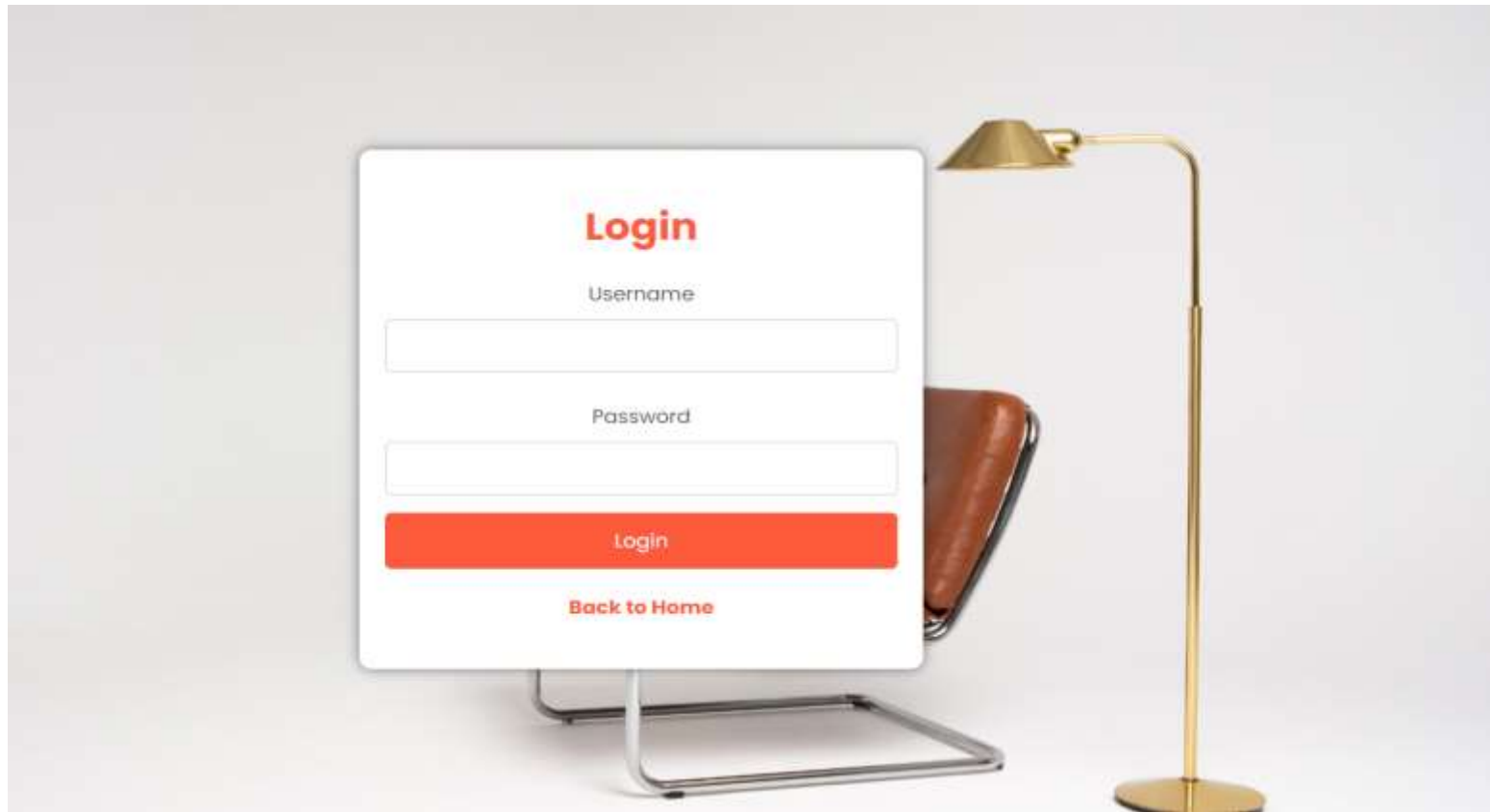


The image shows a digital register form overlaid on a minimalist, modern interior background. The form is white with rounded corners and contains the following elements:

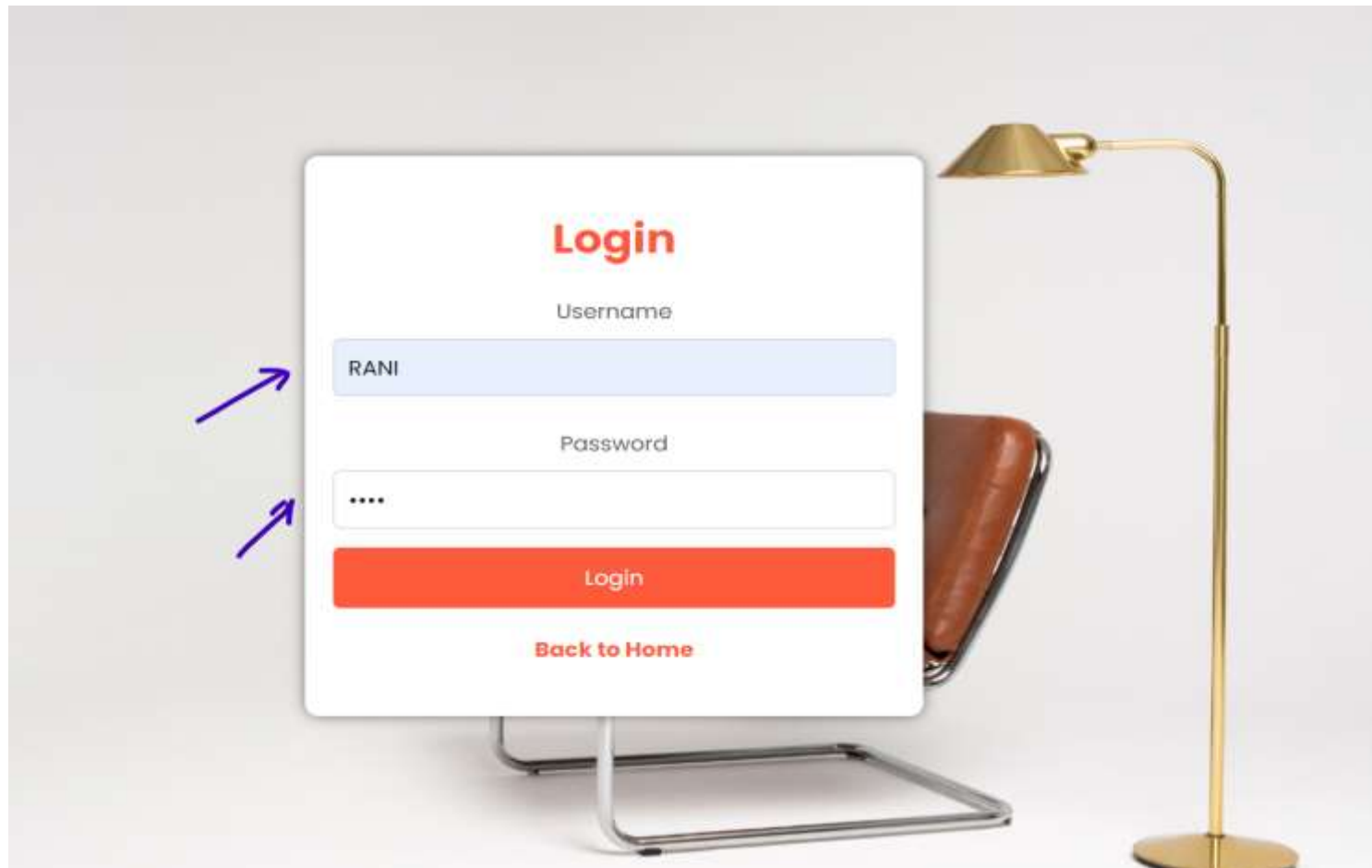
- Register**: A title in bold red text at the top of the form.
- Username**: A label above a white input field.
- Email**: A label above a white input field.
- Password**: A label above a white input field.
- Register**: A solid red button with white text.
- Back to Home**: A red text link at the bottom of the form.

The background features a gold-colored floor lamp and a brown leather chair, suggesting a sophisticated and clean design aesthetic.

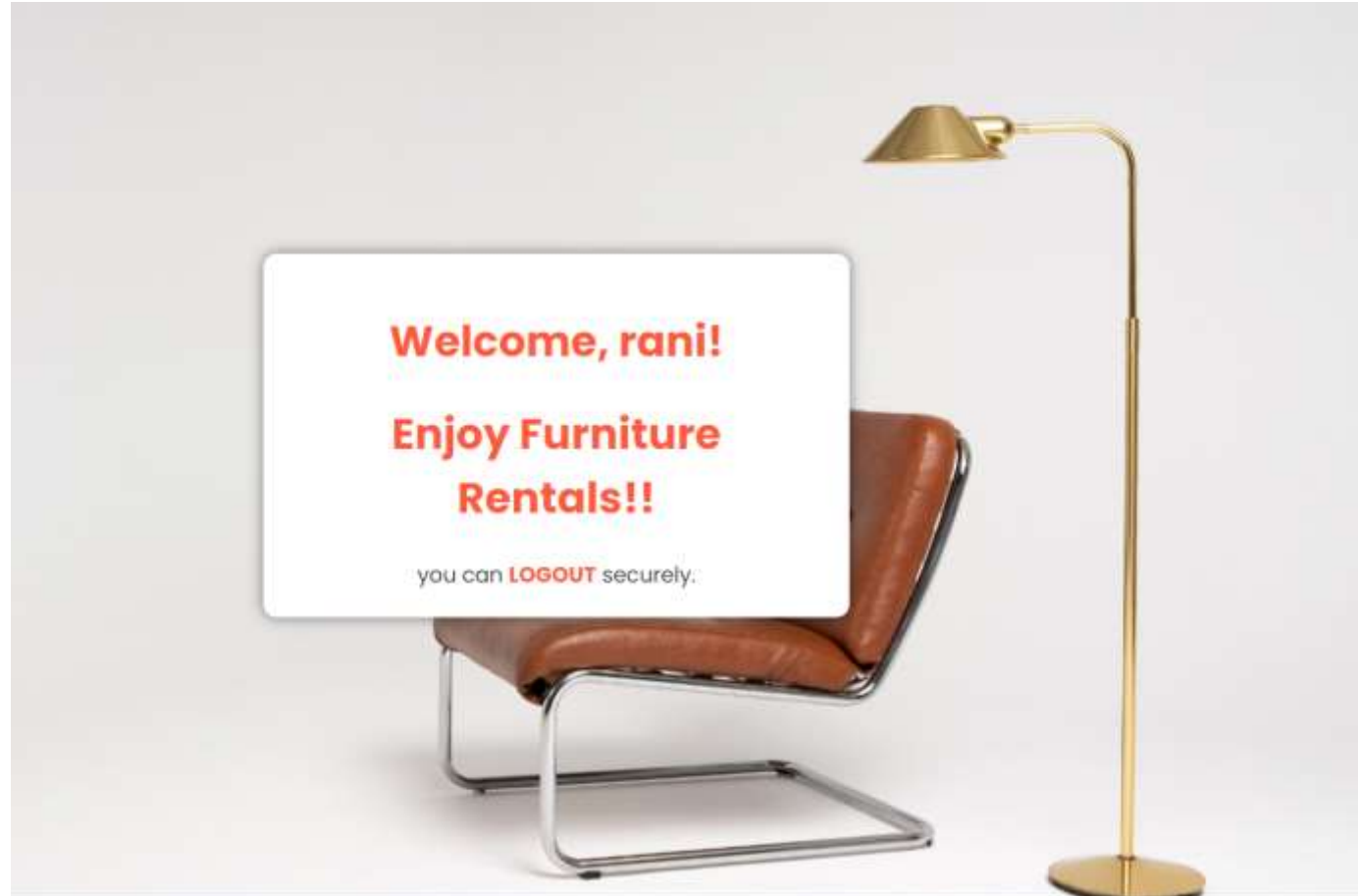
After registering by using the credentials we can login in the application.



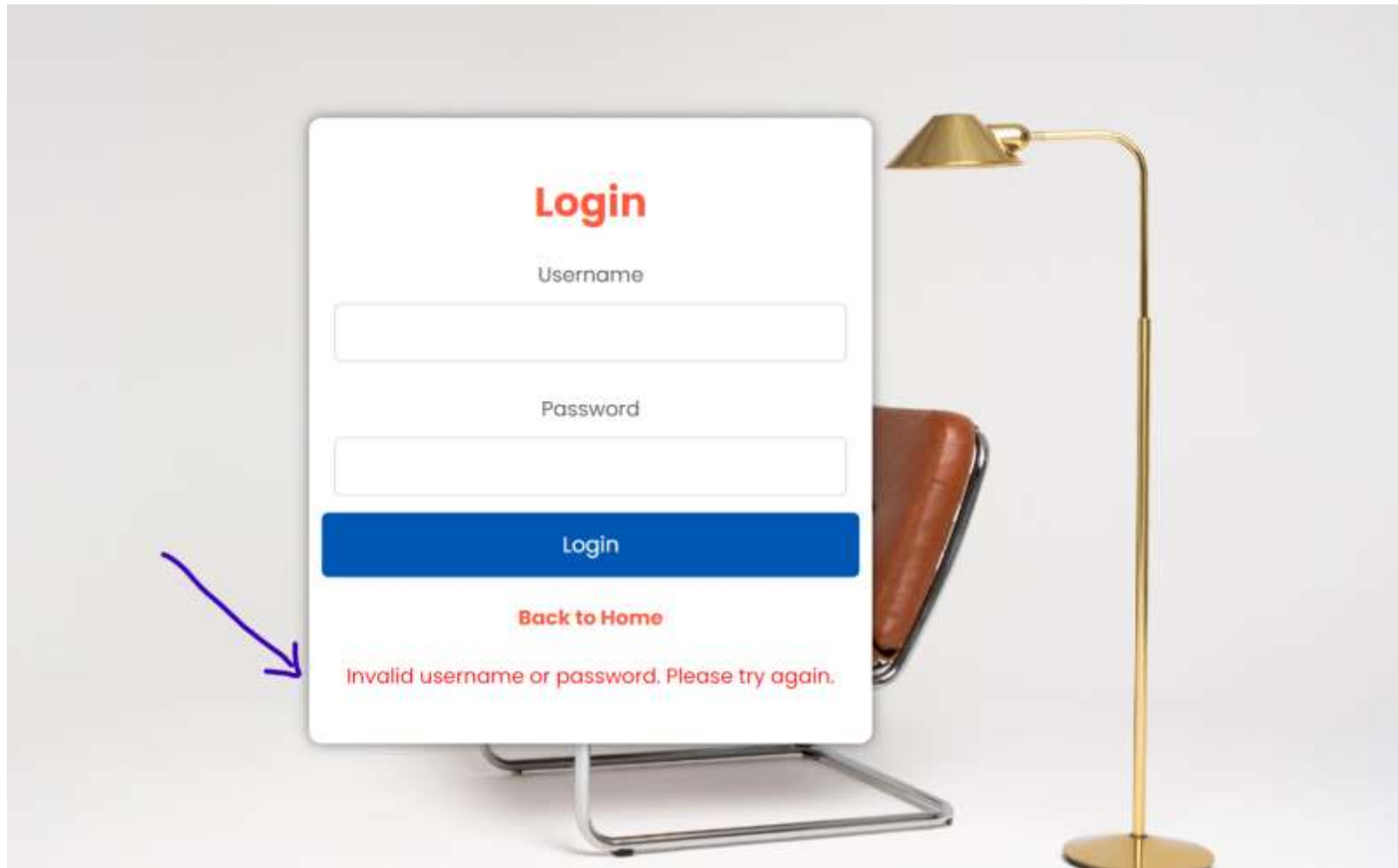
This is how we are giving the credentials



after login in to the page we are seeing a welcome message as shown below



If we are giving any wrong credentials we can see the error message.



Step 2: Required resources

- HTML
- CSS
- JSP
- SERVLETS
- JS
- JDBC
- MYSQL

HTML

- HTML files serve as the backbone of web pages, providing the structure and content that web browsers use to display information on the internet.
- In simple terms, HTML files define the layout, text, images, links, and other elements that make up a webpage, allowing users to interact with and consume content online.
- [Index.html](#)

CSS

- CSS files complement HTML by controlling the presentation and style of web pages. They define how HTML elements should appear on the screen, specifying attributes like colors, fonts, layout, and spacing.
- In simple terms, CSS files make websites visually appealing and user-friendly by applying design rules and enhancing the overall aesthetic of the content presented in HTML files
- `Style.css` and `oldstyle.css`

JSP

- JSP (JavaServer Pages) files are used in web development to dynamically generate web pages. They combine HTML with Java code, allowing developers to create dynamic content that can interact with databases, handle user input, and perform other server-side tasks.
- In simple terms, JSP files enable the creation of dynamic and interactive web applications by embedding Java code within HTML, making it possible to generate customized content based on user input or other factors.
- Login.jsp
- Register.jsp
- Welcome.jsp

SERVLETS

- Servlets are Java classes that extend the capabilities of web servers to generate dynamic web content. They handle requests from web clients (such as web browsers) and produce responses based on the request. Servlets are commonly used to process form data, interact with databases, and perform other server-side tasks in web applications.
- In simple terms, servlets serve as the backbone of dynamic web applications, enabling developers to create interactive and data-driven websites by processing requests and generating appropriate responses.
- `RegisterServlet`
- `loginServlet`

JS

- JavaScript (JS) files are used to add interactivity and dynamic behavior to web pages.
- contain code that can be executed by web browsers to manipulate HTML elements, respond to user actions, and communicate with servers.
- In simple terms, JS files enable developers to create interactive and engaging web experiences by adding features like animations, form validation, interactive maps, and more, enhancing the functionality and user experience of websites.
- Main.jsp

JDBC

- JDBC (Java Database Connectivity) is a Java API that allows programs written in Java to access and manipulate data stored in relational databases. It provides a standard interface for connecting Java applications to database management systems, executing SQL queries, and handling database transactions.
- In simple terms, JDBC enables Java applications to interact with databases, allowing them to store, retrieve, and manipulate data, which is crucial for building database-driven applications such as web applications, enterprise systems, and more.
- DBUtil.java

MYSQL

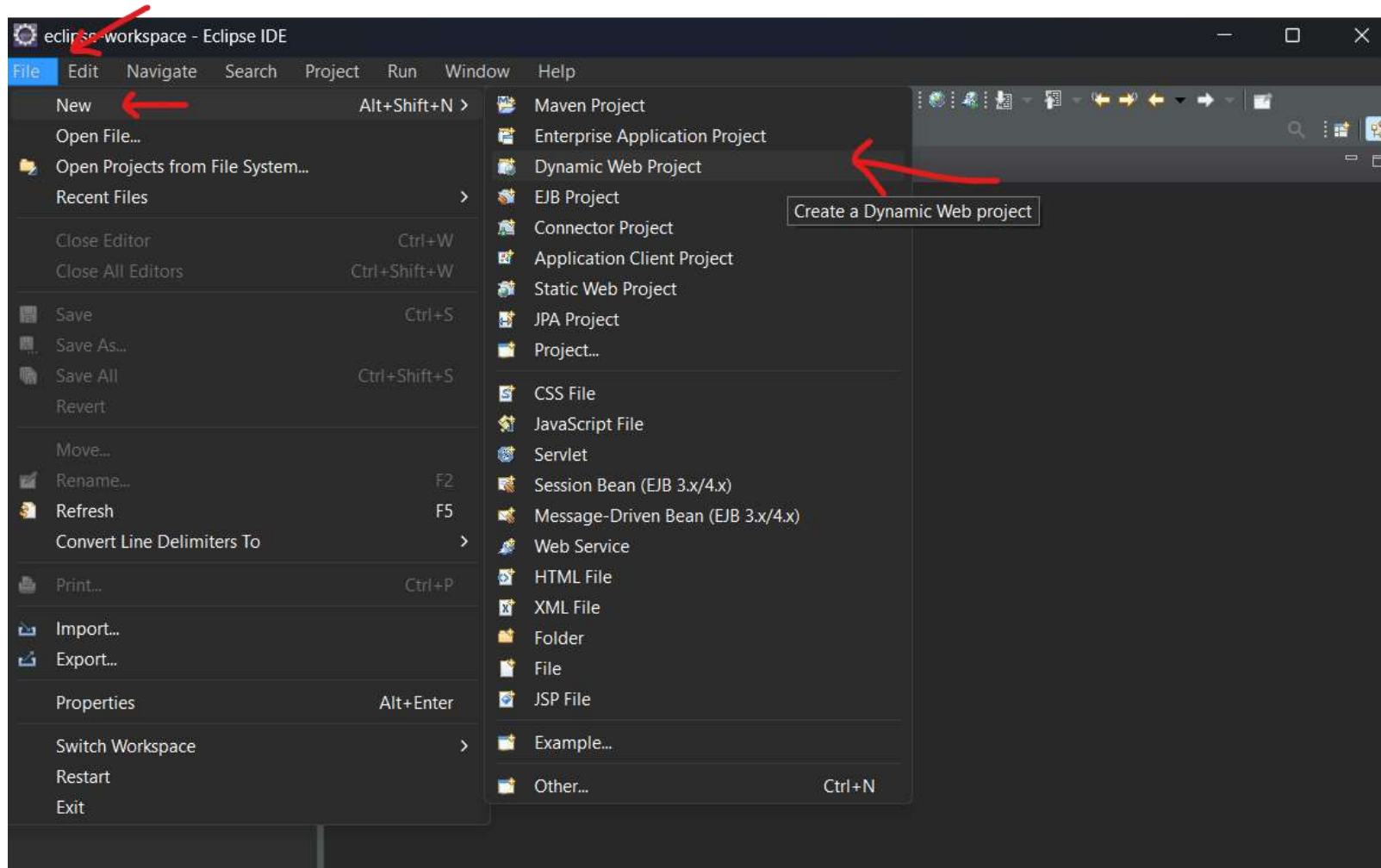
- A "MySQL file" typically refers to the executable file of the MySQL database management system. MySQL is a popular open-source relational database system used to store and manage data in various applications.
- In simple terms, MySQL provides a platform for creating databases, storing data in tables, and performing operations like querying, updating, and deleting data. It's commonly used in web development, business applications, and other software where structured data storage and retrieval are required.
- UserDaoImpl.java
- UserDao.java

FULL STACK PROJECT

VIDEO-2

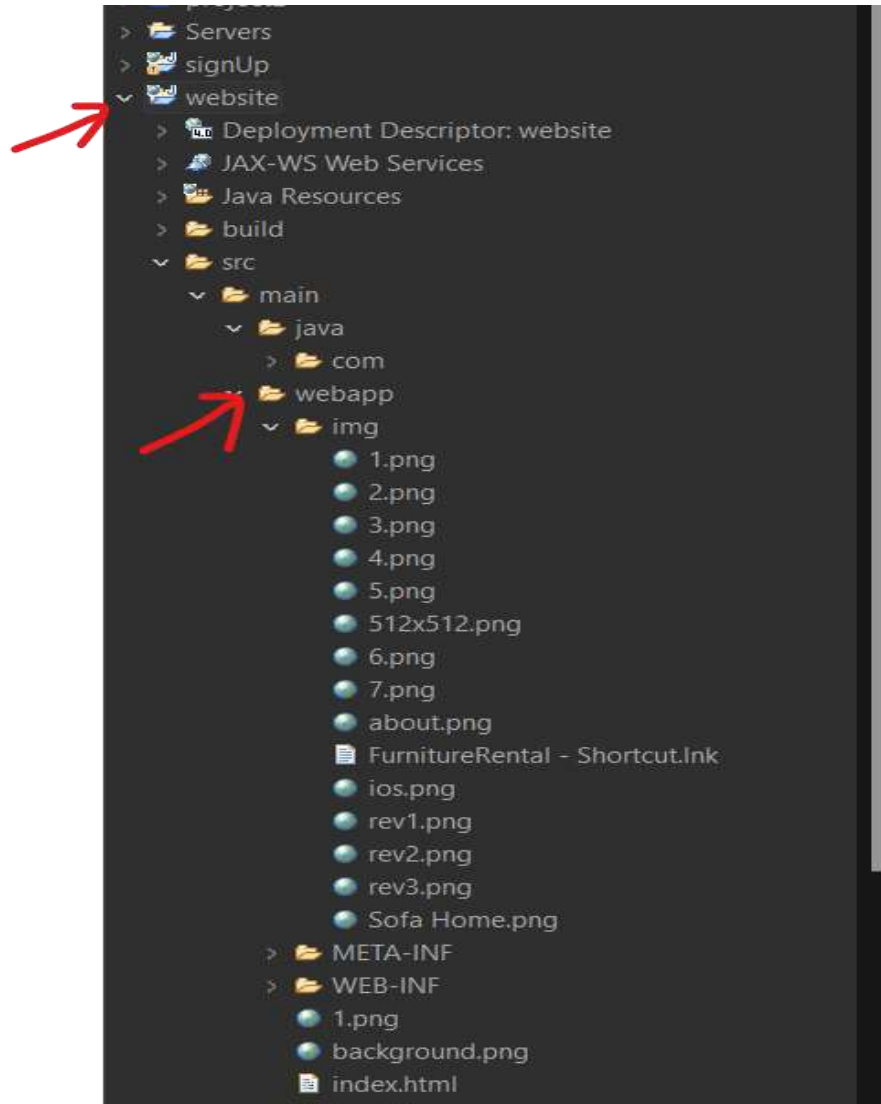
Step 3: Create a dynamic web project

Go to file → new → dynamic web project → click on next → click on next → generate web.xml file → click on finish



Step4: Add a folder of furniture rental website in to the eclipse IDE

Copy the folder and paste it in the project WEBAPP folder.

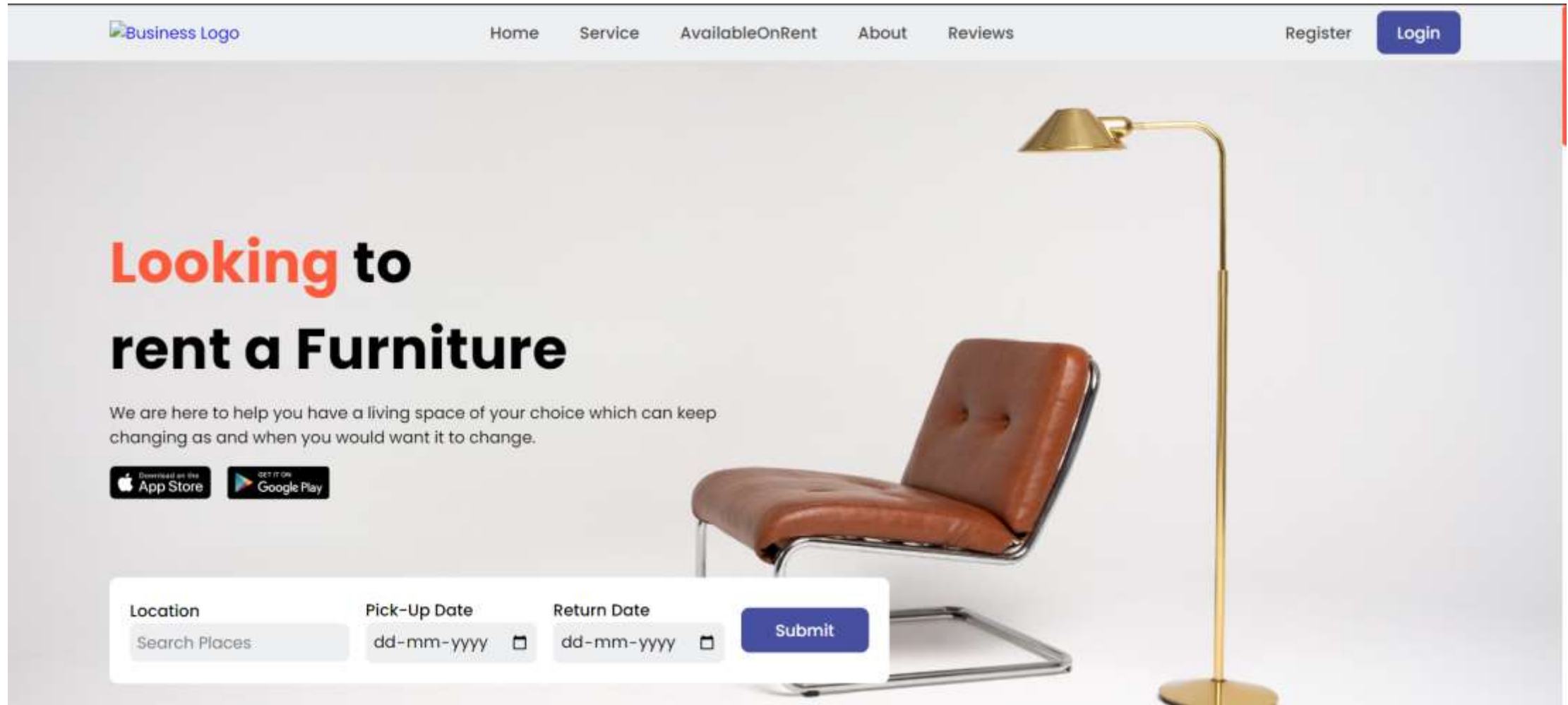


Create index.html file

- Link the register.jsp file and login.jsp file in index.html file.

```
index.html x
1
2 <!DOCTYPE html>
3 <html lang="en">
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Furniture Rental</title>
8
9   <link rel="stylesheet" href="style.css" />
10  <link href='https://unpkg.com/boxicons@2.1.4/css/boxicons.min.css' rel='stylesheet'>
11
12
13 </head>
14 <body>
15
16   <header>
17     <a href="#" class="logo"></a>
18
19     <div class="bx bx-menu" id="menu-icon"></div>
20
21     <ul class="navbar">
22       <li><a href="#home">Home</a></li>
23       <li><a href="#service">Service</a></li>
24       <li><a href="#availableOnRent">AvailableOnRent</a></li>
25       <li><a href="#about">About</a></li>
26       <li><a href="#reviews">Reviews</a></li>
27     </ul>
28
29     <div class="header-btn">
30       <a href="register.jsp" class="sign-up">Register</a>
31       <a href="login.jsp" class="sign-in">Login</a>
32     </div>
33   </header>
34
```

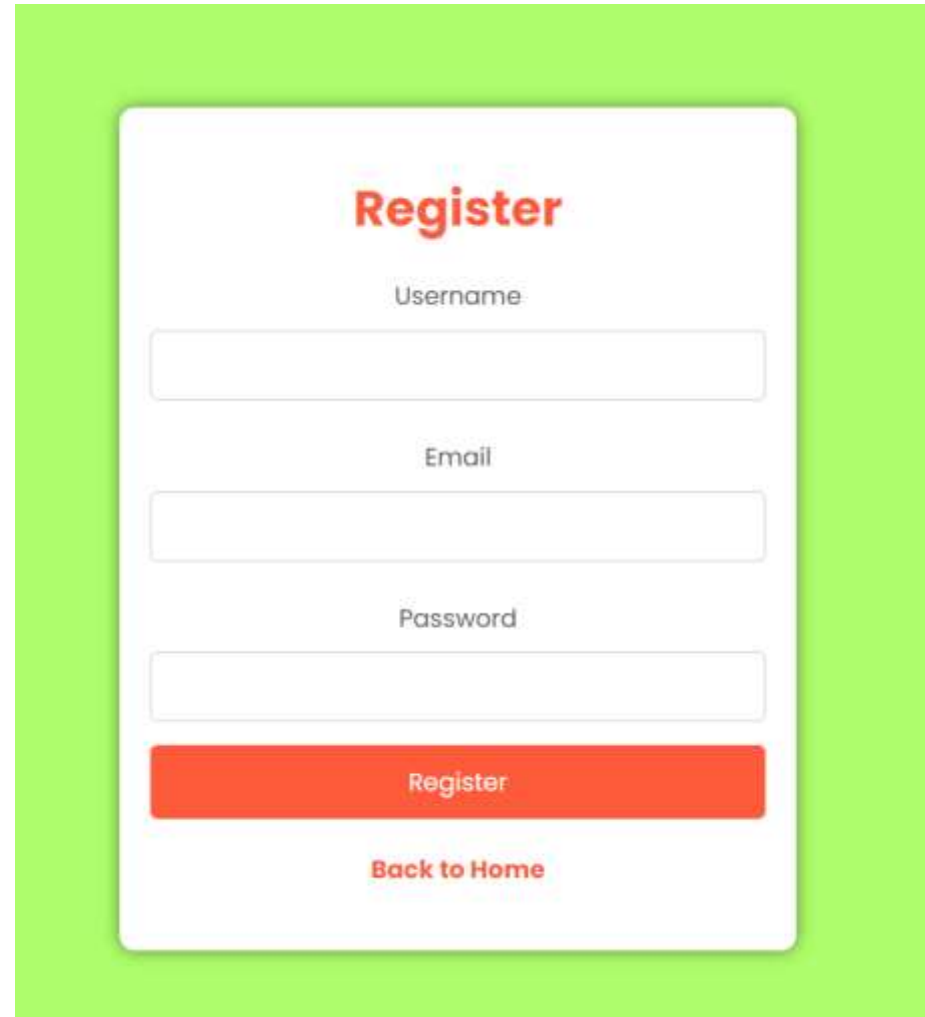
This is how we can see the home page



FULL STACK PROJECT

VIDEO-3

This is our expected Register page



Register

Username

Email

Password

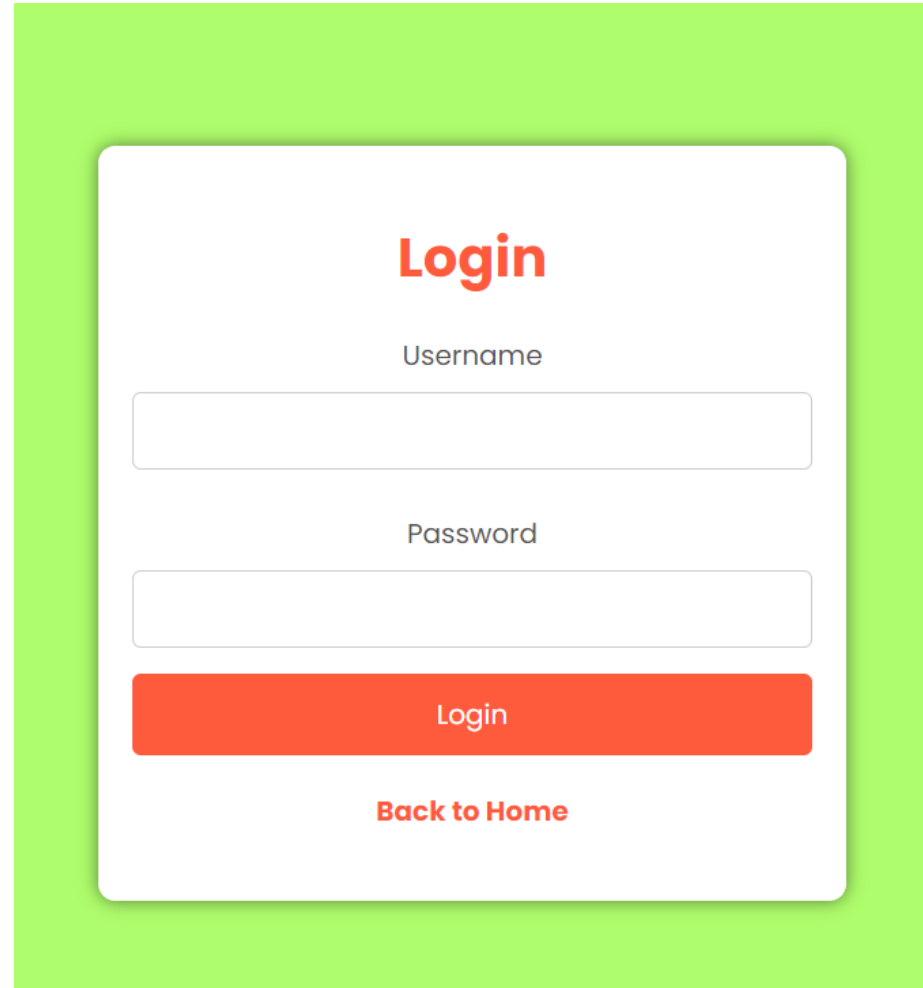
Register

[Back to Home](#)

Create register.jsp file

```
register.jsp x
1 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2   pageEncoding="ISO-8859-1"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6 <meta charset="ISO-8859-1">
7 <title>Register</title>
8 <link rel="stylesheet" type="text/css" href="Oldstyle.css">
9 </head>
10 <body>
11
12 <div class="container">
13   <h1>Register</h1>
14   <form action="RegisterServlet" method="post">
15     <label for="username">Username</label>
16     <input type="text" id="username" name="username" required><br>
17     <label for="email">Email</label>
18     <input type="text" id="username" name="email" required><br>
19     <label for="password">Password</label>
20     <input type="password" id="password" name="password" required><br>
21     <button type="submit">Register</button>
22   </form>
23
24   <p><a href="index.html">Back to Home</a></p>
25 </div>
26 </body>
27 </html>
```

This is our expected Login page



Login

Username

Password

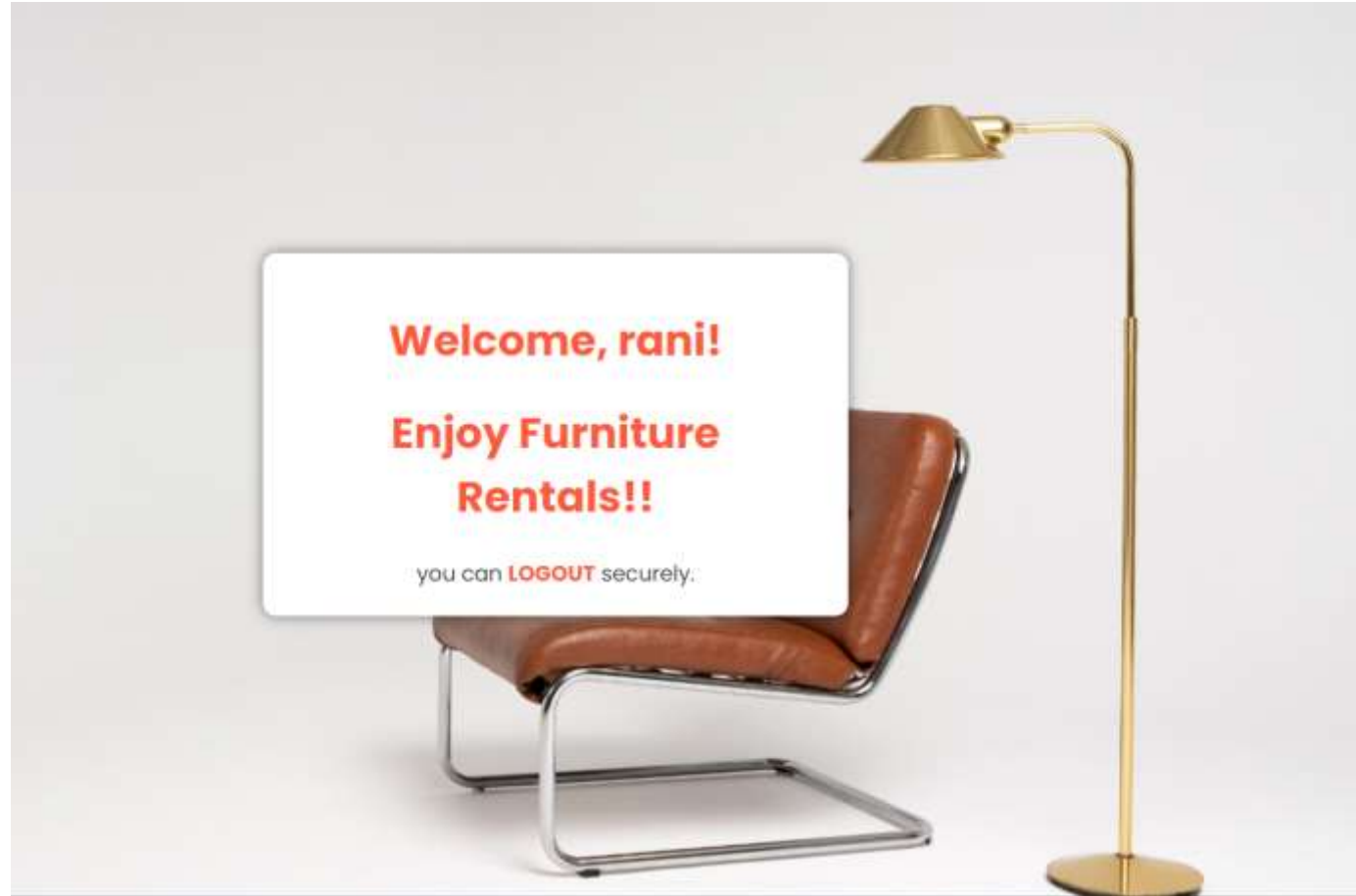
Login

Back to Home

Create login.jsp

```
login.jsp x
1 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2   pageEncoding="ISO-8859-1"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6 <meta charset="ISO-8859-1">
7 <title>Login</title>
8 <link rel="stylesheet" type="text/css" href="Oldstyle.css">
9 </head>
10 <body>
11
12 <div class="container">
13   <h1>Login</h1>
14   <form action="LoginServlet" method="post"> <!-- Change method to "post" -->
15     <label for="username">Username</label>
16     <input type="text" id="username" name="username" required><br>
17     <label for="password">Password</label>
18     <input type="password" id="password" name="password" required><br>
19     <button type="submit">Login</button>
20   </form>
21
22   <p><a href="index.html">Back to Home</a></p>
23   <!-- Display error message if login fails -->
24   <% String error = request.getParameter("error");
25     if (error != null && error.equals("1")) { %>
26     <p style="color: red;">Invalid username or password. Please try again.</p>
27   <% } %>
28   </div>
29 </body>
30 </html>
```

This is our expected welcome message



Create welcome.jsp file

```
welcome.jsp x
1 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2   pageEncoding="ISO-8859-1"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6 <meta charset="ISO-8859-1">
7 <title>Welcome</title>
8 <link rel="stylesheet" type="text/css" href="Oldstyle.css">
9 </head>
10 <body>
11 <body>
12 <%
13     // Retrieve the session object
14     HttpSession session1 = request.getSession(false);
15
16     // Check if the session is not null and the username attribute is set
17     if (session1 != null && session1.getAttribute("username") != null) {
18         // Get the username from the session
19         String username = (String) session1.getAttribute("username");
20     }
21 <div class="container">
22     <h1>Welcome, <%=username%>!</h1>
23     <h1> Enjoy Furniture Rentals!!</h1>
24     you can <a href="Logout.jsp">LOGOUT</a> securely.
25 </div>
26
27
28
29 <%
30 } else {
31     // Redirect to the login page if the session is not valid
32     response.sendRedirect("login.jsp");
33 }
34 %>
35
36 </body>
37 </html>
```

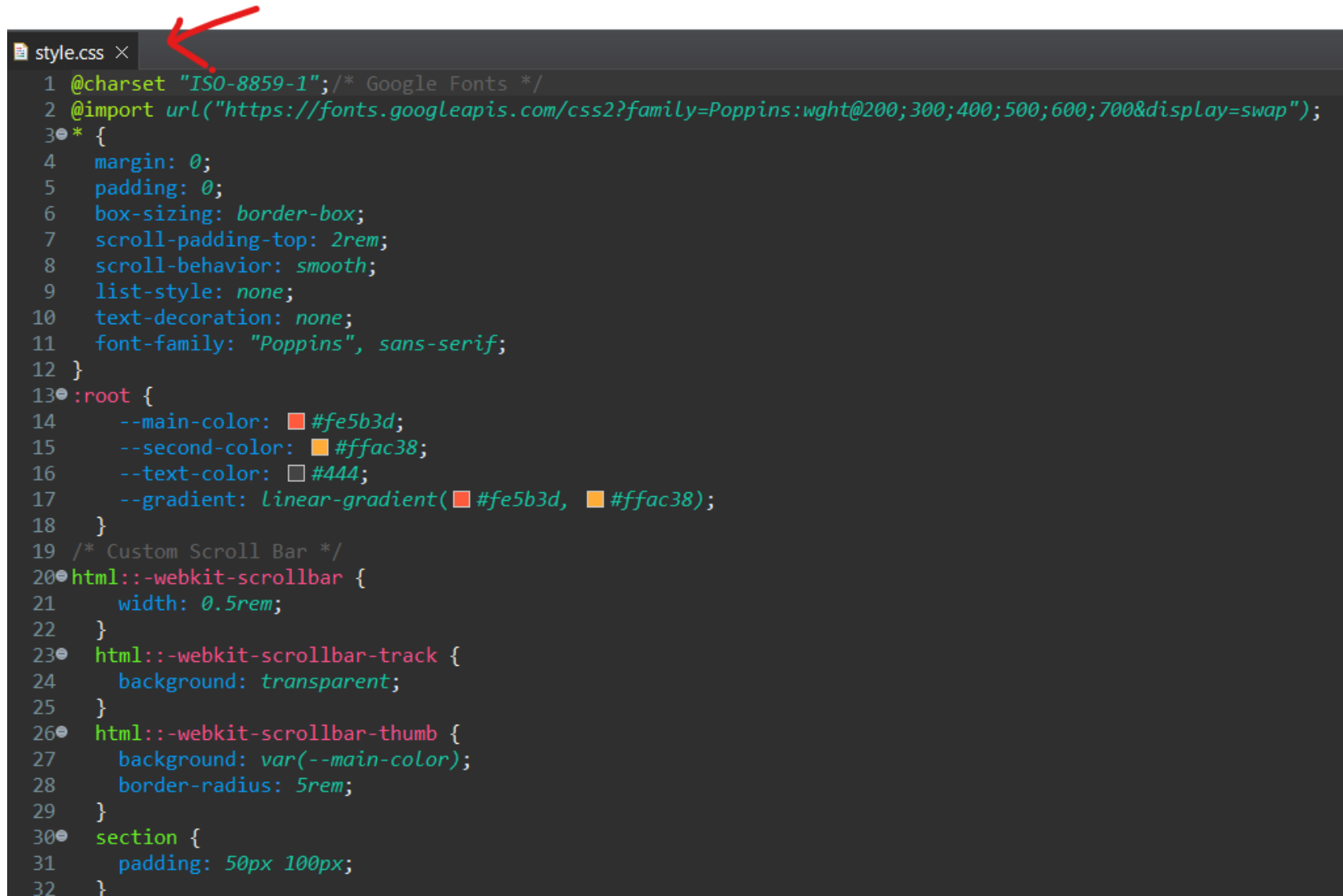
Create logout.jsp file

```
logout.jsp ×
1 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2   pageEncoding="ISO-8859-1"%>
3 <!DOCTYPE html>
4 <html>
5
6 <body>
7 <%
8     // Retrieve the session object
9     HttpSession currentSession = request.getSession(false);
10    if(currentSession != null){
11        currentSession.invalidate();
12    }
13
14
15    // Redirect to the login page if the session is not valid
16    response.sendRedirect("index.html");
17
18    %>
19
20 </body>
21 </html>
```

FULL STACK PROJECT

VIDEO-4

Create one css file to add styles to the home page



```
1 @charset "ISO-8859-1"; /* Google Fonts */
2 @import url("https://fonts.googleapis.com/css2?family=Poppins:wght@200;300;400;500;600;700&display=swap");
3 * {
4   margin: 0;
5   padding: 0;
6   box-sizing: border-box;
7   scroll-padding-top: 2rem;
8   scroll-behavior: smooth;
9   list-style: none;
10  text-decoration: none;
11  font-family: "Poppins", sans-serif;
12 }
13 :root {
14   --main-color: #fe5b3d;
15   --second-color: #ffac38;
16   --text-color: #444;
17   --gradient: linear-gradient(#fe5b3d, #ffac38);
18 }
19 /* Custom Scroll Bar */
20 html::-webkit-scrollbar {
21   width: 0.5rem;
22 }
23 html::-webkit-scrollbar-track {
24   background: transparent;
25 }
26 html::-webkit-scrollbar-thumb {
27   background: var(--main-color);
28   border-radius: 5rem;
29 }
30 section {
31   padding: 50px 100px;
32 }
```

```
style.css x
32
33 header {
34   position: fixed;
35   width: 100%;
36   top: 0;
37   right: 0;
38   z-index: 1000;
39   display: flex;
40   align-items: center;
41   justify-content: space-between;
42   background: #eef1;
43   padding: 15px 100px;
44 }
45 .logo img {
46   width: 40px;
47 }
48 .navbar {
49   display: flex;
50 }
51 .navbar li {
52   position: relative;
53 }
54 .navbar a {
55   font-size: 1rem;
56   padding: 10px 20px;
57   color: var(--text-color);
58   font-weight: 500;
59 }
60 .navbar a::after {
61   content: "";
62   width: 0;
63   height: 3px;
64   background: var(--gradient);
65   position: absolute;
66   bottom: -4px;
67   left: 0;
68   transition: 0.5s;
69 }
```

```
style.css x
70 .navbar a:hover::after {
71   width: 100%;
72 }
73 #menu-icon {
74   font-size: 24px;
75   cursor: pointer;
76   z-index: 10001;
77   display: none;
78 }
79 .header-btn a {
80   padding: 10px 20px;
81   color: var(--text-color);
82   font-weight: 500;
83 }
84 .header-btn .sign-in {
85   background: #474fa0;
86   color: #fff;
87   border-radius: 0.5rem;
88 }
89 .header-btn .sign-in:hover {
90   background: var(--main-color);
91 }
92 .home {
93   width: 100%;
94   min-height: 100vh;
95   position: relative;
96   background: url(img/Sofa\ Home.png);
97   background-repeat: no-repeat;
98   background-position: center right;
99   background-size: cover;
100  display: grid;
101  align-items: center;
102  grid-template-columns: repeat(2, 1fr);
103 }
104 .text h1 {
105   font-size: 3.5rem;
106   letter-spacing: .25px;
107 }
```

```
style.css x
107 }
108 .text span {
109     color: var(--main-color);
110 }
111 .text p {
112     margin: 0.5rem 0 1rem;
113 }
114 .app-stores {
115     display: flex;
116 }
117 .app-stores img {
118     width: 100px;
119     margin-right: 1rem;
120     cursor: pointer;
121 }
122 .form-container form {
123     display: flex;
124     flex-wrap: wrap;
125     align-items: center;
126     gap: 1rem;
127     position: absolute;
128     bottom: 4rem;
129     left: 100px;
130     bottom: 30px;
131     background: #fff;
132     padding: 20px;
133     border-radius: 0.5rem;
134 }
135 }
136 .input-box {
137     flex: 1 1 7rem;
138     display: flex;
139     flex-direction: column;
140 }
141 .input-box span {
142     font-weight: 500;
143 }
```

```
style.css x
144● .input-box input {
145    padding: 7px;
146    outline: none;
147    border: none;
148    background: #eeeeff1;
149    border-radius: 0.5rem;
150    font-size: 1rem;
151  }
152● .form-container form .btn {
153    flex: 1 1 7rem;
154    padding: 10px 34px;
155    border: none;
156    border-radius: 0.5rem;
157    background: #474fa0;
158    color: #fff;
159    font-size: 1rem;
160    font-weight: 500;
161  }
162● .form-container form .btn:hover {
163    background: var(--main-color);
164  }
165● .heading {
166    text-align: center;
167  }
168● .heading span {
169    font-weight: 500;
170    text-transform: uppercase;
171  }
172● .heading h1 {
173    font-size: 2rem;
174  }
175● .service-container {
176    display: grid;
177    align-items: center;
178    grid-template-columns: repeat(auto-fit, minmax(250px, auto));
179    gap: 1rem;
180    margin-top: 2rem;
181  }
```

```
style.css x
181 }
182 .service-container .box {
183   text-align: center;
184   padding: 20px;
185 }
186 .service-container .box .bx {
187   font-size: 34px;
188   padding: 10px;
189   background: #eeeeff1;
190   border-radius: 0.5rem;
191   color: var(--main-color);
192 }
193 .service-container .box h2 {
194   font-size: 1.3rem;
195   font-weight: 500;
196   margin: 1.4rem 0 0.5rem;
197 }
198 .service-container .box .bx:hover,
199 .service-container .box .bx-calendar-check {
200   background: var(--gradient);
201   color: #fff;
202 }
203 .availableOnRent-container {
204   display: grid;
205   grid-template-columns: repeat(auto-fit, minmax(300px, auto));
206   gap: 1rem;
207   margin-top: 2rem;
208 }
209 .availableOnRent-container .box {
210   padding: 10px;
211   border-radius: 1rem;
212   box-shadow: 1px 4px 41px rgba(0, 0, 0, 0.1);
213 }
214 .availableOnRent-container .box .box-img {
215   width: 100%;
216   height: 200px;
217 }
```

```
style.css x
217 }
218 .availableOnRent-container .box .box-img img {
219     width: 100%;
220     height: 100%;
221     border-radius: 1rem;
222     object-fit: cover;
223     object-position: center;
224 }
225 .availableOnRent-container .box p {
226     padding: 0 10px;
227     border: 1px solid var(--text-color);
228     width: 58px;
229     border-radius: 0.5rem;
230     margin: 1rem 0 1rem;
231 }
232 .availableOnRent-container .box h3 {
233     font-weight: 500;
234 }
235 .availableOnRent-container .box h2 {
236     font-size: 1.1rem;
237     font-weight: 600;
238     color: var(--main-color);
239     margin: 0.2rem 0 0.5rem;
240 }
241 .availableOnRent-container .box h2 span {
242     font-size: 0.8rem;
243     font-weight: 500;
244     color: var(--text-color);
245 }
246 .availableOnRent-container .box .btn {
247     display: flex;
248     justify-content: center;
249     background: #474fa0;
250     color: #fff;
251     padding: 10px;
252     border-radius: 0.5rem;
253 }
```

```
style.css x
253 }
254 .availableOnRent-container .box .btn:hover {
255     background: var(--main-color);
256 }
257 .about-container {
258     display: grid;
259     grid-template-columns: repeat(2, 1fr);
260     margin-top: 2rem;
261     align-items: center;
262     gap: 1rem;
263     margin-top: 1rem;
264 }
265 .about-img img {
266     width: 100%;
267 }
268 .about-text span {
269     font-weight: 500;
270     color: var(--main-color);
271     text-transform: uppercase;
272 }
273 .about-text p {
274     margin: 0.5rem 0 1.4rem;
275 }
276 .about-text .btn {
277     padding: 10px 20px;
278     background: #474fa0;
279     color: #fff;
280     border-radius: 0.5rem;
281 }
282 .about-text .btn:hover {
283     background: var(--main-color);
284 }
285 .reviews-container {
286     display: grid;
287     grid-template-columns: repeat(auto-fit, minmax(250px, auto));
288     gap: 1rem;
289     margin-top: 2rem;
290 }
```



```
style.css x
290 }
291 .rev-img {
292   width: 70px;
293   height: 70px;
294 }
295 .rev-img img {
296   width: 100%;
297   height: 100%;
298   border-radius: 50%;
299   object-fit: cover;
300   object-position: center;
301   border: 2px solid var(--second-color);
302 }
303 .reviews-container .box {
304   display: flex;
305   flex-direction: column;
306   align-items: center;
307   text-align: center;
308   padding: 20px;
309   box-shadow: 1px 4px 41px 0px rgba(0, 0, 0, 0.1);
310   border-radius: 0.5rem;
311 }
312 .reviews-container .box h2 {
313   font-size: 1.1rem;
314   font-weight: 600;
315   margin: 0.5rem 0 0.5rem;
316 }
317 .reviews-container .box p {
318   font-style: italic;
319 }
320 .reviews-container .box .stars .bx {
321   color: var(--main-color);
322 }
323 .newsletter {
324   background: linear-gradient(to top right, #fe5b3d, #ffac38);
325   display: flex;
326   flex-direction: column;
327   align-items: center;
328 }
```

```
style.css x
329 .newsletter h2 {
330   color: #fff;
331   font-size: 1.8rem;
332 }
333 .newsletter .box {
334   margin-top: 1rem;
335   background: #fff;
336   border-radius: 0.5rem;
337   padding: 4px 8px;
338   width: 350px;
339   display: flex;
340   justify-content: space-between;
341 }
342 .newsletter .box input {
343   border: none;
344   outline: none;
345 }
346 .newsletter .box .btn {
347   background: #474fa0;
348   color: #fff;
349   padding: 8px 20px;
350   border-radius: 0.5rem;
351 }
352 .copyright {
353   padding: 20px;
354   display: flex;
355   justify-content: space-between;
356   align-items: center;
357 }
358 .social a {
359   color: #444;
360   padding: 10px;
361   font-size: 21px;
362 }
363 /* Making Responsive */
```

```
style.css x
364 @media (max-width: 991px) {
365   header {
366     padding: 18px 40px;
367   }
368   section {
369     padding: 50px 40px;
370   }
371 }
372 @media (max-width: 881px) {
373   .home {
374     background-position: left;
375   }
376   .form-container form {
377     bottom: 0.2rem;
378     left: 40px;
379   }
380 }
381 @media (max-width: 768px) {
382   header {
383     padding: 11px 40px;
384   }
385   #menu-icon {
386     display: initial;
387   }
388   .sign-up {
389     display: none;
390   }
391   .text h1 {
392     font-size: 2.5rem;
393   }
394   .home {
395     grid-template-columns: 1fr;
396   }
397   .form-container form {
398     position: unset;
399   }
```

```
style.css ×
400 header .navbar {
401   position: absolute;
402   top: -500px;
403   left: 0;
404   right: 0;
405   display: flex;
406   flex-direction: column;
407   background: #fff;
408   box-shadow: 0 4px 4px rgba(0, 0, 0, 0.1);
409   transition: 0.2s ease;
410   text-align: left;
411 }
412 .navbar.active {
413   top: 100%;
414 }
415 .navbar a {
416   padding: 1rem;
417   border-left: 2px solid var(--main-color);
418   margin: 1rem;
419   display: block;
420 }
421 .navbar a:hover {
422   color: #fff;
423   background: var(--main-color);
424   border: none;
425 }
426 .navbar a::after {
427   display: none;
428 }
429 .heading span {
430   font-size: 0.9rem;
431   font-weight: 600;
432 }
433 .heading h1 {
434   font-size: 1.3rem;
435 }
```

```
style.css x
435 }
436 .about-container {
437     grid-template-columns: 1fr;
438     text-align: center;
439 }
440 .about-img {
441     padding: 1rem;
442     order: 2;
443 }
444 }
445 @media (max-width: 568px) {
446     .copyright {
447         flex-direction: column;
448     }
449     .newsletter .box {
450         width: 284px;
451     }
452     .form-container {
453         padding-top: 2rem;
454     }
455 }
456 @media (max-width: 350px) {
457     header {
458         padding: 4px 14px;
459     }
460     .logo img {
461         width: 30px;
462     }
463     section {
464         padding: 50px 14px;
465     }
466     .header-btn .sign-in {
467         padding: 7px 10px;
468         font-size: 14px;
469         font-weight: 400;
470     }
471     .text h1 {
472         font-size: 2rem;
473     }
474     .availableOnRent-container {
475         grid-template-columns: repeat(auto-fit, minmax(254px, auto));
476     }
477 }
```

Create another css file to add styles to the login, register and welcome pages



```
Oldstyle.css ×
1 @charset "ISO-8859-1";
2 @import url("https://fonts.googleapis.com/css2?family=Poppins:wght@200;300;400;500;600;700&display=swap");
3 * {
4   font-family: "Poppins", sans-serif;
5 }
6 body {
7   display: flex;
8   flex-direction: column;
9   align-items: center;
10  justify-content: center;
11  height: 100vh;
12  margin: 0;
13  background-color: #AEFD6C;
14  background-image: url('1.png');
15  background-size: cover; /* or 'contain' based on your preference */
16  background-position: center center;
17  background-repeat: no-repeat;
18 }
19 }
20
21 .container {
22   width: 80%;
23   max-width: 400px;
24   padding: 20px;
25   border-radius: 10px;
26   box-shadow: 0 0 10px rgba(0, 0, 1, 0.5);
27   background-color: #AEFD6C;
28   text-align: center; /* Center the content */
29 }
30
31 h1 {
32   font-weight: bold;
33   text-align: center;
34   color: #333;
35 }
36
```

```
Oldstyle.css ×
31●h1 {
32    font-weight: bold;
33    text-align: center;
34    color: █ #333;
35 }
36
37●label {
38    display: block;
39    margin: 10px 0;
40    color: █ #555;
41 }
42
43●input {
44    width: 100%;
45    padding: 10px;
46    margin-bottom: 15px;
47    box-sizing: border-box;
48    border: 1px solid █ #ccc;
49    border-radius: 5px;
50    font-size: 16px;
51 }
52
53●button {
54    width: 100%;
55    padding: 12px;
56    cursor: pointer;
57    background-color: █ #fe5b3d;
58    color: █ #fff;
59    border: none;
60    border-radius: 5px;
61    font-size: 16px;
62    transition: background-color 0.3s ease, transform 0.3s ease;
63 }
64
65●button:hover {
66    background-color: █ #0056b3;
67    transform: scale(1.05);
68 }
```

```
Oldstyle.css ×
43 input {
44     width: 100%;
45     padding: 10px;
46     margin-bottom: 15px;
47     box-sizing: border-box;
48     border: 1px solid #ccc;
49     border-radius: 5px;
50     font-size: 16px;
51 }
52
53 button {
54     width: 100%;
55     padding: 12px;
56     cursor: pointer;
57     background-color: #fe5b3d;
58     color: #fff;
59     border: none;
60     border-radius: 5px;
61     font-size: 16px;
62     transition: background-color 0.3s ease, transform 0.3s ease;
63 }
64
65 button:hover {
66     background-color: #0056b3;
67     transform: scale(1.05);
68 }
69
70 a {
71     text-decoration: none;
72     color: #fe5b3d;
73     font-weight: bold;
74 }
75
76 a:hover {
77     color: #0056b3;
78 }
79
80 /*@charset "ISO-8859-1";*/
```



```
Oldstyle.css x
79
80 /*@charset "ISO-8859-1";*/
81 body {
82     font-family: 'Arial', sans-serif;
83     display: flex;
84     flex-direction: column;
85     align-items: center;
86     justify-content: center;
87     height: 100vh;
88     margin: 0;
89     background-color: #AEFD6C;
90     background-image: url('1.png');
91     background-size: cover;
92     background-position: center center;
93     background-repeat: no-repeat;
94     color: #333;
95 }
96
97 .container {
98     width: 600px;
99     padding: 20px;
100     border-radius: 10px;
101     box-shadow: 0 0 10px rgba(0, 0, 1, 0.5);
102     background-color: white;
103     text-align: center;
104 }
105
106 h1 {
107     font-weight: bold;
108     color: #fe5b3d;
109 }
110
111 p {
112     margin: 20px 0;
113     line-height: 1.6;
114 }
115
```

```
114 }  
115  
116 .emoji {  
117     font-size: 24px;  
118     margin-right: 5px;  
119 }  
120  
121  
122 .links a {  
123     margin: 0 10px;  
124     text-decoration: none;  
125     color:  #fe5b3d;  
126     font-weight: bold;  
127     transition: color 0.3s ease;  
128 }  
129  
130 .links a:hover {  
131     color:  #fe5b3d;  
132     font-size: 20px;  
133     text-transform: uppercase;  
134 }  
135
```

Create main.js file

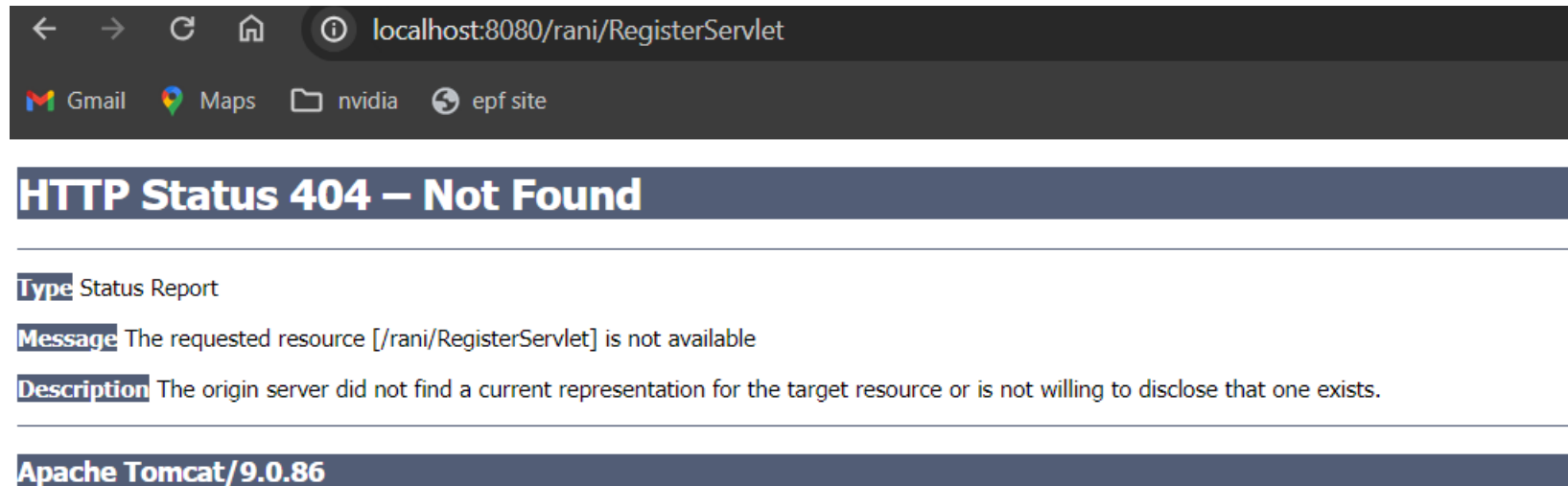
```
main.js ×
1 let menu = document.querySelector('#menu-icon');
2 let navbar = document.querySelector('.navbar');
3
4 menu.onclick = () => {
5     menu.classList.toggle('bx-x');
6     navbar.classList.toggle('active');
7 }
8
9 window.onscroll = () => {
10     menu.classList.remove('bx-x');
11     navbar.classList.remove('active');
12 }
13
14 const sr = ScrollReveal ({
15     distance: '60px',
16     duration: 2500,
17     delay: 400,
18     reset: true
19 })
20
21 sr.reveal('.text',{delay: 200, origin: 'top'})
22 sr.reveal('.form-container form',{delay: 800, origin: 'left'})
23 sr.reveal('.heading',{delay: 800, origin: 'top'})
24 sr.reveal('.service-container .box',{delay: 600, origin: 'top'})
25 sr.reveal('.availableOnRent-container .box',{delay: 600, origin: 'top'})
26 sr.reveal('.about-container .box',{delay: 600, origin: 'top'})
27 sr.reveal('.reviews-container',{delay: 600, origin: 'top'})
28 sr.reveal('.newsletter .box',{delay: 400, origin: 'bottom'})
```

FULL STACK PROJECT

VIDEO-5

- Till now we have worked on front end development.
- So we have created a home page, register page, login page.
- Let us try to register by giving the credentials.

Till now we have not yet given any dynamic action to the register page. For that we need to create a registerServlet class



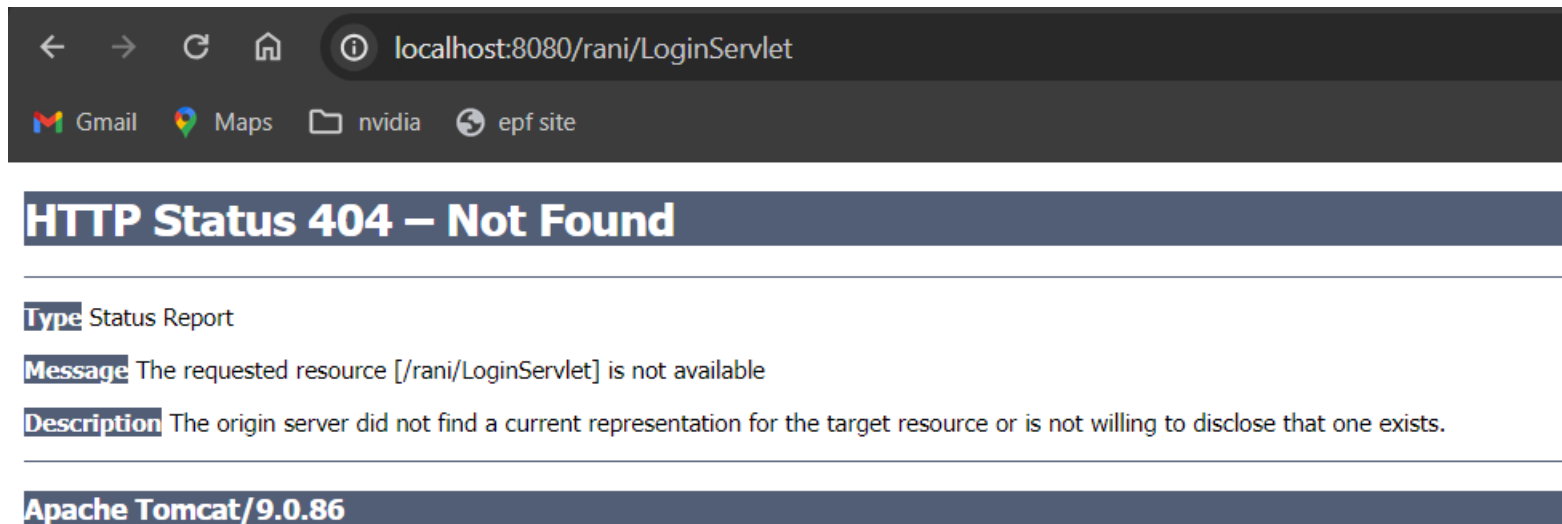
Create RegisterServlet.file

```
LoginServlet.java  RegisterServlet.java ×
1 package com.company.servlet;
2
3 import java.io.IOException;
4 import javax.servlet.ServletException;
5 import javax.servlet.annotation.WebServlet;
6 import javax.servlet.http.HttpServlet;
7 import javax.servlet.http.HttpServletRequest;
8 import javax.servlet.http.HttpServletResponse;
9
10 import com.company.dao.UserDaoImpl;
11 import com.company.model.User;
12
13 /**
14  * Servlet implementation class RegisterServlet
15  */
16 @WebServlet("/RegisterServlet")
17 public class RegisterServlet extends HttpServlet {
18     private static final long serialVersionUID = 1L;
19     private static UserDaoImpl userDao = new UserDaoImpl();
20
21     /**
22      * @see HttpServlet#HttpServlet()
23      */
24     public RegisterServlet() {
25         super();
26         // TODO Auto-generated constructor stub
27     }
28
29
30     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
```

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
  
    String username = request.getParameter("username");  
    String password = request.getParameter("password");  
    String email = request.getParameter("email");  
  
    User user = new User();  
    user.setUsername(username);  
    user.setEmail(email);  
    user.setPassword(password);  
  
    // UserDao userDao = new UserDaoImpl();  
  
    if (userDao.addUser((User) user)) {  
        response.sendRedirect("login.jsp?registration=success");  
    } else {  
        response.sendRedirect("register.jsp?error=1");  
    }  
}  
}
```


- We have created register page.
- Try to register by giving the username, email id and create a password.
- After registering try to login the login page.

We have not yet created a LoginServlet class



Create LoginServlet.file

```
LoginServlet.java ×
1 package com.company.servlet;
2
3 import java.io.IOException;
4 import javax.servlet.ServletException;
5 import javax.servlet.annotation.WebServlet;
6 import javax.servlet.http.HttpServlet;
7 import javax.servlet.http.HttpServletRequest;
8 import javax.servlet.http.HttpServletResponse;
9 import javax.servlet.http.HttpSession;
10
11 import com.company.dao.UserDaoImpl;
12
13 /**
14  * Servlet implementation class LoginServlet
15  */
16 @WebServlet("/LoginServlet")
17 public class LoginServlet extends HttpServlet {
18     private static final long serialVersionUID = 1L;
19     private static UserDaoImpl userdao = new UserDaoImpl();
20
21     /**
22      * @see HttpServlet#HttpServlet()
23      */
24     public LoginServlet()
25     {
26         super();
27         // TODO Auto-generated constructor stub
28     }
29
30     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
```

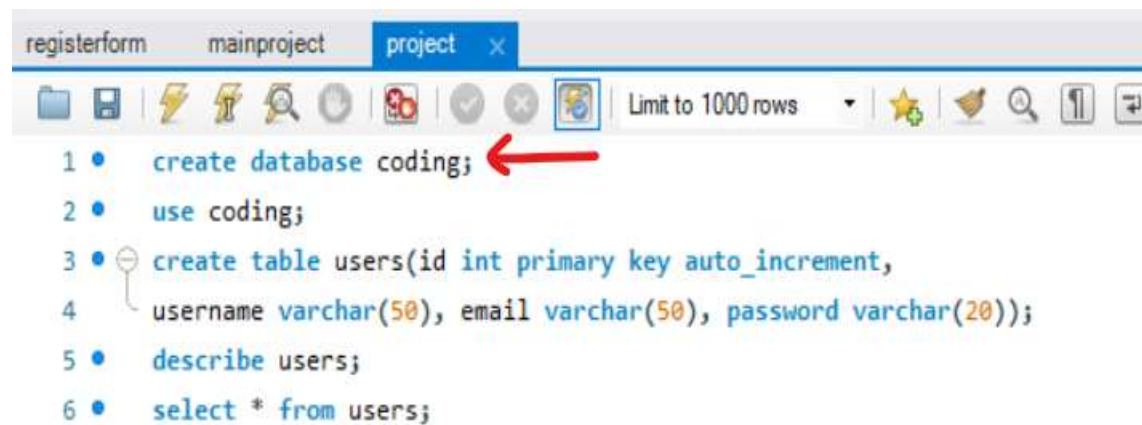
```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
    // TODO Auto-generated method stub  
    String username = request.getParameter("username");  
    String password = request.getParameter("password");  
  
    if (userdao.isValidUser(username, password)) {  
        HttpSession session = request.getSession();  
        session.setAttribute("username", username);  
        response.sendRedirect("welcome.jsp");  
        // System.out.println("Hi - "+username);  
    }  
  
    else {  
        response.sendRedirect("login.jsp?error=1");  
        System.out.println("Error");  
    }  
  
}  
}
```

FULL STACK PROJECT

VIDEO-6

step 5 create a database


I have created a database with the name of coding. To communicate with this database we need to add the mysql connector to the eclipse IDE



```
1 • create database coding;
2 • use coding;
3 • create table users(id int primary key auto_increment,
4 • username varchar(50), email varchar(50), password varchar(20));
5 • describe users;
6 • select * from users;
```


- Go to the browser type mysql connector 8.0.32.jar

Product version: 8.3.0, 8.2.0, 8.1.0, 8.0.33, 8.0.32, 8.0.31, 8.0.30, 8.0.29 ...

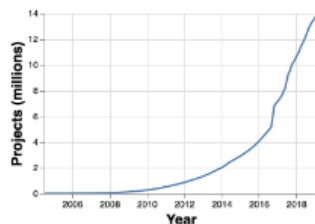
 **Maven Repository**
https://mvnrepository.com › artifact › com.mysql › 8.0.32

[mysql-connector-j » 8.0.32](#) ✓

18 Jan 2023 — **MySQL Connector/J** is a JDBC Type 4 driver, which means that it is pure Java implementation of the MySQL protocol and does not rely on the ...

 **MySQL**
https://downloads.mysql.com › archives › c-net

Indexed Artifacts (35.5M)



Popular Categories

Testing Frameworks & Tools

Android Packages

Logging Frameworks

Java Specifications

JSON Libraries

JVM Languages

Language Runtime

Core Utilities

Mocking

Web Assets

Annotation Libraries

HTTP Clients

Home » [com.mysql](#) » [mysql-connector-j](#) » 8.0.32

MySQL Connector/J » 8.0.32

MySQL Connector/J is a JDBC Type 4 driver, which means that it is pure Java implementation of the MySQL protocol and does the MySQL client libraries. This driver supports auto-registration with the Driver Manager, standardized validity checks, category SQLExceptions, support for large update counts, support for local and offset date-time variants from the java.time package, support for JDBC-4.x XML processing, support for per connection client information and support for the NCHAR, NVARCHAR ...

Categories	JDBC Drivers
Tags	database sql jdbc driver connector rdbms mysql connection
Organization	Oracle Corporation
HomePage	http://dev.mysql.com/doc/connector-j/en/
Date	Jan 18, 2023
Files	pom (3 KB) jar (2.4 MB) View All
Repositories	Central
Ranking	#787 in MvnRepository (See Top Artifacts) #9 in JDBC Drivers
Used By	630 artifacts

Note: There is a new version for this artifact

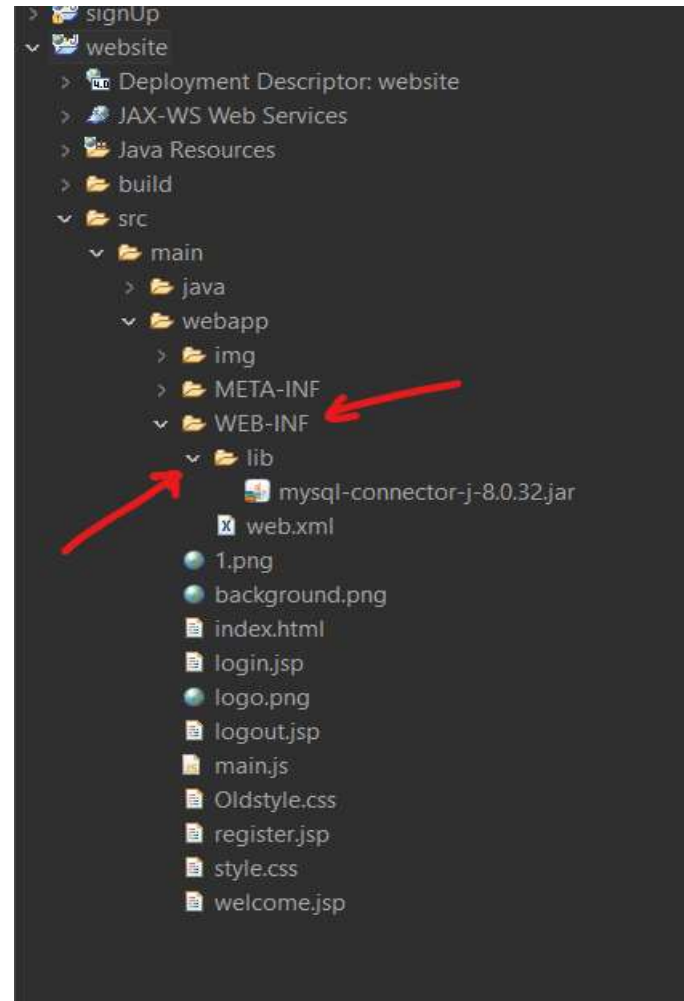
New Version

8.4.0

com/mysql/mysql-connector-j/8.0.32

../		
mysql-connector-j-8.0.32-javadoc.jar	2023-01-18 16:04	1091416
mysql-connector-j-8.0.32-javadoc.jar.asc	2023-01-18 16:04	836
mysql-connector-j-8.0.32-javadoc.jar.md5	2023-01-18 16:04	32
mysql-connector-j-8.0.32-javadoc.jar.sha1	2023-01-18 16:04	40
mysql-connector-j-8.0.32-sources.jar	2023-01-18 16:04	1607222
mysql-connector-j-8.0.32-sources.jar.asc	2023-01-18 16:04	836
mysql-connector-j-8.0.32-sources.jar.md5	2023-01-18 16:04	32
mysql-connector-j-8.0.32-sources.jar.sha1	2023-01-18 16:04	40
mysql-connector-j-8.0.32.jar	2023-01-18 16:04	2480823
mysql-connector-j-8.0.32.jar.asc	2023-01-18 16:04	836
mysql-connector-j-8.0.32.jar.md5	2023-01-18 16:04	32
mysql-connector-j-8.0.32.jar.sha1	2023-01-18 16:04	40
mysql-connector-j-8.0.32.pom	2023-01-18 16:04	3180
mysql-connector-j-8.0.32.pom.asc	2023-01-18 16:04	836
mysql-connector-j-8.0.32.pom.md5	2023-01-18 16:04	32
mysql-connector-j-8.0.32.pom.sha1	2023-01-18 16:04	40

- Copy that downloaded mysql connect and paste it in to the project in WEB-INF under lib folder



Step:6 Establish the connection to the database we need to create a class called DBUtil

```
UserDaoImpl.java  UserDao.java  User.java  DBUtil.java ×
1 package com.company.util;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.SQLException;
6
7 public class DBUtil {
8     private static final String URL = "jdbc:mysql://localhost:3306/coding";
9     private static final String USERNAME = "root";
10    private static final String PASSWORD = "Chinni@27";
11    static {
12        try {
13            // Load the MySQL JDBC driver
14            Class.forName("com.mysql.cj.jdbc.Driver");
15        } catch (ClassNotFoundException e) {
16            e.printStackTrace();
17        }
18    }
19
20    public static Connection getConnection() throws SQLException {
21        return DriverManager.getConnection(URL, USERNAME, PASSWORD);
22    }
23 }
24
```

Interact with the database create a class called UserDaoImpl

```
UserDaoImpl.java ×
1 package com.company.dao;
2
3 import java.sql.Connection;
4 import java.sql.PreparedStatement;
5 import java.sql.ResultSet;
6 import java.sql.SQLException;
7
8 import com.company.model.User;
9 import com.company.util.DBUtil;
10
11 public class UserDaoImpl implements UserDao {
12     @Override
13     public boolean addUser(User user) {
14         String query = "INSERT INTO users (username, email, password) VALUES (?, ?, ?)";
15
16         try (Connection connection = DBUtil.getConnection();
17             PreparedStatement preparedStatement = connection.prepareStatement(query)) {
18
19             preparedStatement.setString(1, user.getUsername());
20             preparedStatement.setString(2, user.getEmail());
21             preparedStatement.setString(3, user.getPassword());
22
23             int rowsAffected = preparedStatement.executeUpdate();
24
25             return rowsAffected > 0;
26         } catch (SQLException e) {
27             e.printStackTrace();
28             return false;
29         }
30     }
31 }
```

```
    }  
  
    @Override  
    public boolean isValidUser(String username, String password) {  
        String query = "SELECT * FROM users WHERE username = ? AND password = ?";  
        try (Connection connection = DBUtil.getConnection();  
            PreparedStatement preparedStatement = connection.prepareStatement(query)) {  
  
            preparedStatement.setString(1, username);  
            preparedStatement.setString(2, password);  
  
            ResultSet resultSet = preparedStatement.executeQuery();  
  
            return resultSet.next();  
        } catch (SQLException e) {  
            e.printStackTrace();  
            return false;  
        }  
    }  
}
```

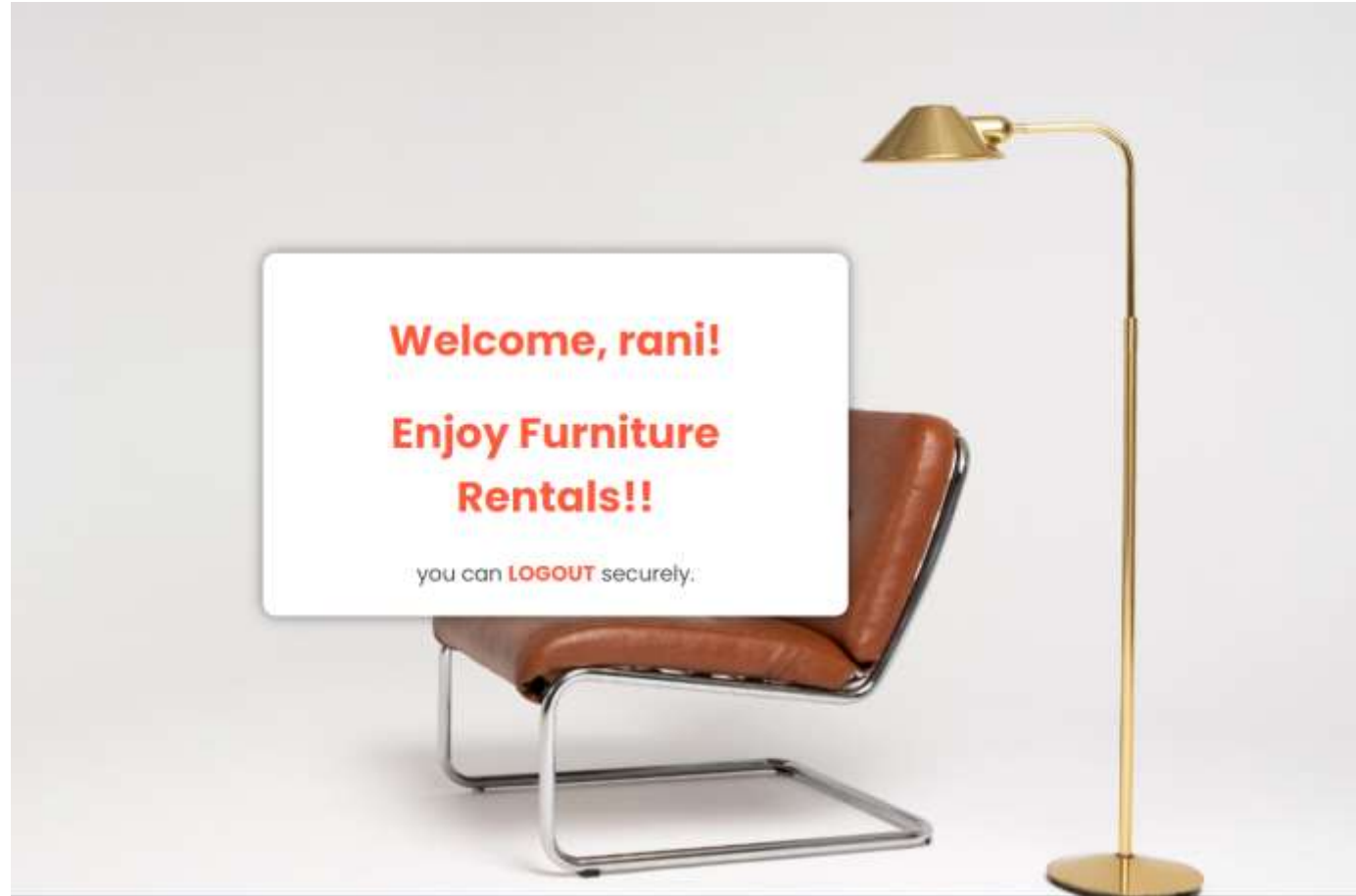
Create a java interface called UserDao for handling the user data in a system.

```
UserDao.java ×
1 package com.company.dao;
2
3 import com.company.model.User;
4
5 public interface UserDao {
6
7     boolean addUser(User user);
8     boolean isValidUser(String username, String password);
9
10 }
```

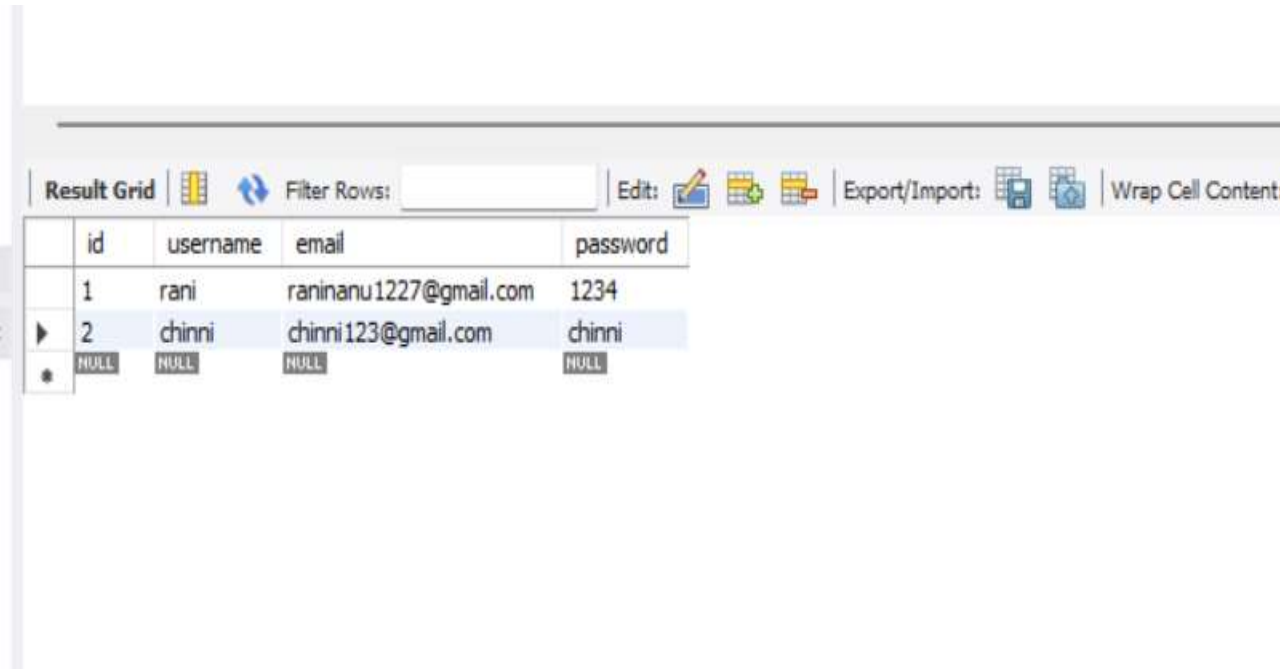
- Create a class called User.java this class helps to provides a basic structure to represent a user in a system. It is used to encapsulates the users data(username, password and email)

```
User.java X
1 package com.company.model;
2
3 public class User {
4     private String username;
5     private String password;
6     private String email;
7     public String getUsername() {
8         return username;
9     }
10    public void setUsername(String username) {
11        this.username = username;
12    }
13    public String getEmail() {
14        return email;
15    }
16    public void setEmail(String email) {
17        this.email = email;
18    }
19    public String getPassword() {
20        return password;
21    }
22    public void setPassword(String password) {
23        this.password = password;
24    }
25
26
27
28 }
```


Final output



And this is how user data is stored in database



The screenshot shows a database management interface with a 'Result Grid' tab. The grid displays a table with four columns: 'id', 'username', 'email', and 'password'. The first row contains the values '1', 'rani', 'raninanu1227@gmail.com', and '1234'. The second row, which is highlighted, contains '2', 'chinni', 'chinni123@gmail.com', and 'chinni'. Below the second row, there is a row with four 'NULL' values. The interface also includes a 'Filter Rows' field, an 'Edit' button, and an 'Export/Import' button.

	id	username	email	password
	1	rani	raninanu1227@gmail.com	1234
▶	2	chinni	chinni123@gmail.com	chinni
*	NULL	NULL	NULL	NULL

All the best