

Daily Assignment: 6-11/ November/2025

Backend:

The screenshot shows the Eclipse IDE interface with the following components visible:

- Package Explorer:** Shows the project structure for "jwt-auth-backend [boot] [devtools]". The `src/main/java` package contains several sub-packages and files:
 - `com.example.jwtauthbackend`: Contains `JwtAuthBackendApplication.java`.
 - `com.example.jwtauthbackend.config`: Contains `SecurityConfig.java`.
 - `com.example.jwtauthbackend.controller`: Contains `AuthController.java`.
 - `com.example.jwtauthbackend.filter`: Contains `JwtFilter.java` and `MockUserDetailsService.java`.
 - `com.example.jwtauthbackend.util`: Contains `AuthRequest.java`, `AuthResponse.java`, and `JwtUtil.java`.
- Java Editor:** Displays the contents of `JwtAuthBackendApplication.java`. The code is as follows:

```
1 package c
2
3 import or
4 import or
5 import or
6
7 @SpringBo
8 @EnableWe
9 public cl
10 public
11     s
12 }
13 }
```

- Boot Dashboard:** Shows the project status with a green power icon and the text "local". It also displays a search bar and a message: "Type tags, projects, or working set names to match (incl. *)".
- Problems View:** Shows a single entry: "jwt-auth-backen".

Jwt auth backend application.java:

The screenshot shows a Java IDE interface with multiple tabs at the top. The active tab is 'JwtAuthBack...' containing the following code:

```
1 package com.example.jwtauthbackend;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
6
7 @SpringBootApplication
8 @EnableWebSecurity
9 public class JwtAuthBackendApplication {
10     public static void main(String[] args) {
11         SpringApplication.run(JwtAuthBackendApplication.class, args);
12     }
13 }
```

Security config.java

```
14 import org.springframework.security.web.SecurityFilterChain;
15 import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;
16
17 @Configuration
18 @EnableMethodSecurity
19 public class SecurityConfig {
20
21     @Autowired
22     private JwtFilter jwtFilter;
23
24     @Bean
25     public PasswordEncoder passwordEncoder() {
26         return new BCryptPasswordEncoder();
27     }
28
29     @Bean
30     public AuthenticationManager authenticationManager(AuthenticationConfiguration config) throws Exception {
31         return config.getAuthenticationManager();
32     }
33
34     @Bean
35     public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
36         http.csrf(csrf -> csrf.disable())
37             .authorizeHttpRequests(
38                 auth -> auth.requestMatchers("/auth/login").permitAll().anyRequest().authenticated())
39             .sessionManagement(session -> session.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
40             .addFilterBefore(jwtFilter, UsernamePasswordAuthenticationFilter.class);
41
42     return http.build();
43 }
44 }
```

Authcontroller.java

```

6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.http.ResponseEntity;
8 import org.springframework.security.access.prepost.PreAuthorize;
9 import org.springframework.web.bind.annotation.*;
10
11 @RestController
12 @RequestMapping("/auth")
13 @CrossOrigin(origins = "*")
14 public class AuthController {
15
16     @Autowired
17     private JwtUtil jwtUtil;
18
19     private static final String VALID_USERNAME = "student";
20     private static final String VALID_PASSWORD = "password123";
21
22     @PostMapping("/login")
23     public ResponseEntity<?> createAuthenticationToken(@RequestBody AuthRequest authenticationRequest)
24             throws Exception {
25         String username = authenticationRequest.getUsername();
26         String password = authenticationRequest.getPassword();
27
28         if (username.equals(VALID_USERNAME) && password.equals(VALID_PASSWORD)) {
29             final String token = jwtUtil.generateToken(username);
30             return ResponseEntity.ok(new AuthResponse(token, "Login successful"));
31         } else {
32             return ResponseEntity.status(401).body("Invalid credentials");
33         }
34     }
35
36     @GetMapping("/profile")
37     @PreAuthorize("hasRole('USER')")
38     public ResponseEntity<?> getProfile() {
39         String username = "student";
40         return ResponseEntity.ok("Welcome, " + username + "! You are authenticated.");
41     }
42 }

```

Jwtfilter.java:

```

J JwtAuthBack... SecurityConf... JwtFilter.java X JwtUtil.java AuthControll...
1/ import java.io.IOException;
18
19 @Component
20 public class JwtFilter extends OncePerRequestFilter {
21
22     @Autowired
23     private JwtUtil jwtUtil;
24
25     @Autowired
26     private UserDetailsService userDetailsService;
27
28     @Override
29     protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain)
30             throws ServletException, IOException {
31         String authorizationHeader = request.getHeader("Authorization");
32
33         String token = null;
34         String username = null;
35
36         if (authorizationHeader != null && authorizationHeader.startsWith("Bearer ")) {
37             token = authorizationHeader.substring(7);
38             try {
39                 username = jwtUtil.extractUsername(token);
40             } catch (Exception e) {
41             }
42         }
43
44         if (username != null && SecurityContextHolder.getContext().getAuthentication() == null) {
45             UserDetails userDetails = userDetailsService.loadUserByUsername(username);
46
47             if (jwtUtil.validateToken(token, userDetails.getUsername())) {
48                 UsernamePasswordAuthenticationToken authToken = new UsernamePasswordAuthenticationToken(userDetails,
49                         null, userDetails.getAuthorities());
50                 authToken.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));
51                 securityContextHolder.getContext().setAuthentication(authToken);
52             }
53         }
54
55         filterChain.doFilter(request, response);
56     }

```

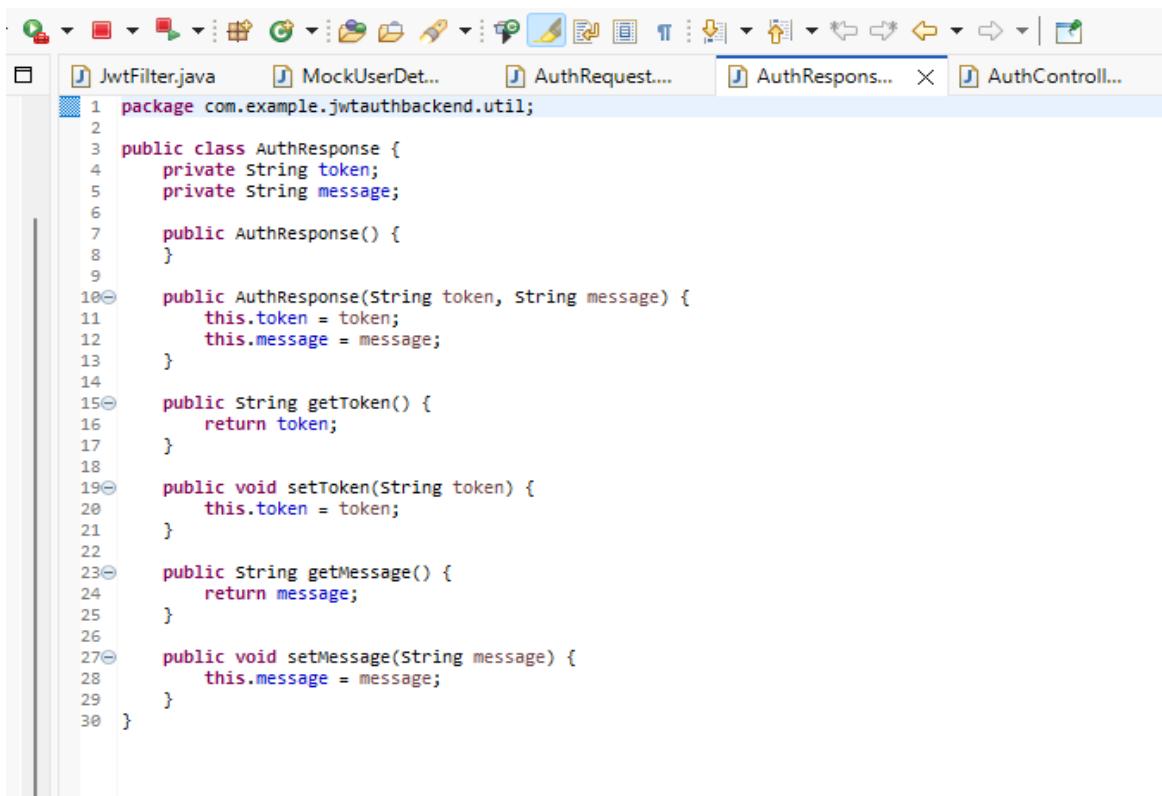
MockuserDetailsService.java

```
1 package com.example.jwtauthbackend.filter;
2
3 import org.springframework.security.core.userdetails.User;
4 import org.springframework.security.core.userdetails.UserDetails;
5 import org.springframework.security.core.userdetails.UserDetailsService;
6 import org.springframework.security.core.userdetails.UsernameNotFoundException;
7 import org.springframework.stereotype.Service;
8
9 @Service
10 public class MockUserDetailsService implements UserDetailsService {
11
12     @Override
13     public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
14         if ("student".equals(username)) {
15             return User.withUsername(username).password("{noop}password123").authorities("ROLE_USER")
16                     .accountExpired(false).accountLocked(false).credentialsExpired(false).disabled(false).build();
17         } else {
18             throw new UsernameNotFoundException("User not found: " + username);
19         }
20     }
21 }
```

Authrequest.java

```
1 package com.example.jwtauthbackend.util;
2
3 public class AuthRequest {
4     private String username;
5     private String password;
6
7     public AuthRequest() {
8     }
9
10    public AuthRequest(String username, String password) {
11        this.username = username;
12        this.password = password;
13    }
14
15    public String getUsername() {
16        return username;
17    }
18
19    public void setUsername(String username) {
20        this.username = username;
21    }
22
23    public String getPassword() {
24        return password;
25    }
26
27    public void setPassword(String password) {
28        this.password = password;
29    }
30 }
```

Authresponse.java



The screenshot shows a Java IDE interface with the following details:

- Toolbar:** Standard Java IDE toolbar with icons for file operations, search, and code navigation.
- Project Explorer:** Shows several Java files: JwtFilter.java, MockUserDet..., AuthRequest..., AuthRespons..., and AuthControll... (partially visible).
- Code Editor:** The AuthResponse.java file is open in the editor. The code defines a class with private fields token and message, and methods to get and set these values.

```
1 package com.example.jwtauthbackend.util;
2
3 public class AuthResponse {
4     private String token;
5     private String message;
6
7     public AuthResponse() {
8     }
9
10    public AuthResponse(String token, String message) {
11        this.token = token;
12        this.message = message;
13    }
14
15    public String getToken() {
16        return token;
17    }
18
19    public void setToken(String token) {
20        this.token = token;
21    }
22
23    public String getMessage() {
24        return message;
25    }
26
27    public void setMessage(String message) {
28        this.message = message;
29    }
30 }
```

Jwtutil.java

The screenshot shows an IDE interface with two code editors and several toolbars.

Top Editor:

```

1 package com.example.jwtauthbackend.util;
2
3 import io.jsonwebtoken.Claims;
4 import io.jsonwebtoken.Jwts;
5 import io.jsonwebtoken.security.Keys;
6 import io.jsonwebtoken.security.SignatureException;
7 import org.springframework.stereotype.Component;
8
9 import javax.crypto.SecretKey;
10 import java.util.Date;
11 import java.util.HashMap;
12 import java.util.Map;
13 import java.util.function.Function;
14
15 @Component
16 public class JwtUtil {
17     private final SecretKey secretKey = Keys.secretKeyFor(io.jsonwebtoken.SignatureAlgorithm.HS256);
18     private final long EXPIRATION_TIME = 3600000;
19
20     public String extractUsername(String token) {
21         return extractClaim(token, Claims::getSubject);
22     }
23
24     public Date extractExpiration(String token) {
25         return extractClaim(token, Claims::getExpiration);
26     }
27
28     public <T> T extractClaim(String token, Function<Claims, T> claimsResolver) {
29         final Claims claims = extractAllClaims(token);
30         return claimsResolver.apply(claims);
31     }
32
33     private Claims extractAllClaims(String token) {
34         return Jwts.parser().verifyWith(secretKey).build().parseSignedClaims(token).getPayload();
35     }
36
37     private Boolean isTokenExpired(String token) {
38         return extractExpiration(token).before(new Date());
39     }

```

Bottom Editor:

```

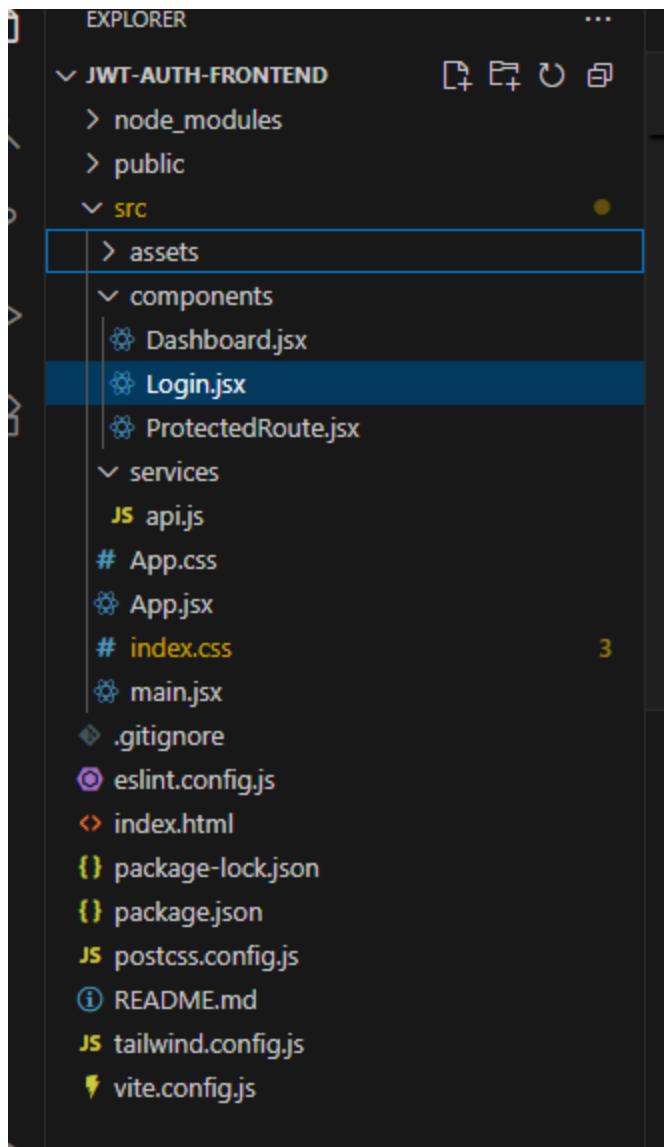
24     public Date extractExpiration(String token) {
25         return extractClaim(token, Claims::getExpiration);
26     }
27
28     public <T> T extractClaim(String token, Function<Claims, T> claimsResolver) {
29         final Claims claims = extractAllClaims(token);
30         return claimsResolver.apply(claims);
31     }
32
33     private Claims extractAllClaims(String token) {
34         return Jwts.parser().verifyWith(secretKey).build().parseSignedClaims(token).getPayload();
35     }
36
37     private Boolean isTokenExpired(String token) {
38         return extractExpiration(token).before(new Date());
39     }
40
41     public String generateToken(String username) {
42         Map<String, Object> claims = new HashMap<>();
43         return createToken(claims, username);
44     }
45
46     private String createToken(Map<String, Object> claims, String subject) {
47         return Jwts.builder().claims(claims).subject(subject).issuedAt(new Date(System.currentTimeMillis()))
48             .expiration(new Date(System.currentTimeMillis() + EXPIRATION_TIME)).signWith(secretKey).compact();
49     }
50
51     public Boolean validateToken(String token, String username) {
52         try {
53             final String extractedUsername = extractUsername(token);
54             return (extractedUsername.equals(username) && !isTokenExpired(token));
55         } catch (SignatureException | io.jsonwebtoken.ExpiredJwtException | io.jsonwebtoken.MalformedJwtException e) {
56             return false; // Invalid token
57         } catch (Exception e) {
58             return false;
59         }
60     }
61 }

```

Right Side Panels:

- Task List:** Shows a single entry: "Find".
- Outline:** Shows the class structure of `JwtUtil`, including methods like `secretKey`, `EXPIRATION_TIME`, `extractUsername`, `extractExpiration`, `extractClaim`, `extractAllClaims`, `isTokenExpired`, `generateToken`, and `createToken`.

Frontend:



Dashboard.jsx

```
JWT-authFrontend
```

```
# App.css      Login.jsx    Dashboard.jsx  # index.css 3
```

```
src > components > Dashboard.jsx > ...
```

```
1 import React, { useState, useEffect } from "react";
2 import api from "../services/api";
3
4 const decodeJWT = (token) => {
5   try {
6     const payload = JSON.parse(atob(token.split(".")[1]));
7     return payload.exp * 1000;
8   } catch {
9     return null;
10 }
11 };
12
13 const Dashboard = () => {
14   const [profile, setProfile] = useState("");
15   const [remainingTime, setRemainingTime] = useState(0);
16   const [user] = useState({ name: "Student User", email: "student@example.com" });
17
18   const handleLogout = () => {
19     localStorage.removeItem("token");
20     window.location.href = "/login";
21   };
22
23   useEffect(() => {
24     const fetchProfile = async () => {
25       try {
26         const res = await api.get("/auth/profile");
27         setProfile(res.data);
28       } catch {
29         setProfile("Authenticated successfully!");
30       }
31     };
32     fetchProfile();
33
34     const token = localStorage.getItem("token");
35     if (token) {
36       const expTime = decodeJWT(token);
37       if (expTime) {
38         const interval = setInterval(() => {
39           const remaining = Math.floor((expTime - Date.now()) / 1000);
40           setRemainingTime(remaining);
41           if (remaining <= 0) {
42             clearInterval(interval);
43             handleLogout();
44           }
45         }, 1000);
46         return () => clearInterval(interval);
47       }
48     }
49   
```

```
# App.css      Login.jsx    Dashboard.jsx  # index.css 3
```

```
src > components > Dashboard.jsx > ...
```

```
13 const Dashboard = () => {
14   useEffect(() => {
15     handleLogout();
16   }, []);
17
18   const formatTime = (s) =>
19     `${Math.floor(s / 60)}:${String(s % 60).padStart(2, "0")}`;
20
21   return (
22     <div className="min-h-screen relative flex flex-col text-white overflow-hidden">
23       <!-- Blue gradient background -->
24       <div className="absolute inset-0 bg-gradient-to-br from-[#0a2342] via-[#002e5d] to-[#0066a1]" />
25       <div className="absolute inset-0 backdrop-blur-2xl opacity-80" />
26
27       <!-- Decorative blur circles -->
28       <div className="absolute top-10 left-10 w-72 h-72 bg-cyan-400 rounded-full filter blur-[120px] opacity-20" />
29       <div className="absolute bottom-10 right-10 w-72 h-72 bg-blue-500 rounded-full filter blur-[120px] opacity-20" />
30
31       <!-- Header -->
32       <header className="relative z-10 flex justify-between items-center p-8 max-w-6xl mx-auto w-full">
33         <h1 className="text-3xl font-semibold bg-gradient-to-r from-cyan-400 via-blue-300 to-blue-500 bg-clip-text text-transparent">
34           Welcome, <span className="font-bold">{user.name}</span>
35         </h1>
36         <button
37           onClick={handleLogout}
38           className="px-5 py-2 bg-gradient-to-r from-cyan-500 to-blue-600 hover:from-cyan-400 hover:to-blue-500 text-white rounded-lg shadow-lg transition-transform transform"
39         >Logout
40         </button>
41       </header>
42
43       <!-- Dashboard Grid -->
44       <main className="relative z-10 flex flex-col justify-center items-center px-6">
45         <div className="grid grid-cols-3 md:grid-cols-3 gap-6 w-full max-w-6xl">
46           <!-- Profile Card -->
47           <div className="bg-white/10 backdrop-blur-x1 border border-white/20 rounded-2xl p-6 shadow-2xl hover:shadow-blue-500/20 transition-all">
48             <h3 className="text-xl font-semibold mb-2 text-cyan-300">
49               Profile
50             </h3>
51             <p className="text-gray-200 mb-2">{profile}</p>
52             <p className="text-sm text-gray-300">Email: {user.email}</p>
53           </div>
54
55           <!-- Session Timer -->
56           <div className="bg-white/10 backdrop-blur-x1 border border-white/20 rounded-2xl p-6 text-center shadow-2xl hover:shadow-blue-500/20 transition-all">
57             <h3 className="text-xl font-semibold mb-2 text-cyan-300">
58               Session Timer
59             </h3>
60           </div>
61         </div>
62       </main>
63     </div>
64   
```

```

src > components > @ Dashboard.jsx ...
13  const Dashboard = () => {
91    <h3 className="text-xl font-semibold mb-2 text-cyan-300">
92      Session Timer
93    </h3>
94    <p>
95      className={["text-4xl font-bold ${",
96      `remainingTime < 300 ? "text-red-400" : "text-green-400"`,
97      `"}]}
98      >
99      | {formatTime(remainingTime)}
100     </p>
101     <p className="text-gray-300 mt-2 text-sm">
102       Token expiration countdown
103     </p>
104   </div>
105
106  /* Quick Actions */
107  <div className="bg-white/10 backdrop-blur-xl border border-white/20 rounded-2xl p-6 shadow-2xl hover:shadow-blue-500/20 transition-all">
108    <h3 className="text-xl font-semibold mb-2 text-cyan-300">
109      Quick Actions
110    </h3>
111    <div className="flex gap-4 mt-3">
112      <button className="px-4 py-2 bg-gradient-to-r from-cyan-500 to-blue-600 rounded-xl text-sm font-semibold hover:from-cyan-400 hover:to-blue-500 transition">
113        View Reports
114      </button>
115      <button className="px-4 py-2 bg-gradient-to-r from-cyan-500 to-blue-600 rounded-xl text-sm font-semibold hover:from-cyan-400 hover:to-blue-500 transition">
116        Edit Profile
117      </button>
118    </div>
119  </div>
120 </div>
121 </main>
122
123  /* Footer */
124  <footer className="relative z-10 text-center py-6 text-gray-300 text-sm">
125    © 2025 JWT Auth App - Built with ❤️ using React + Spring Boot
126  </footer>
127 </div>
128 </>;
129 </>;
130 </>;
131 export default Dashboard;
132 |

```

Login.jsx

```

# App.css          # Login.jsx X  # Dashboard.jsx  # index.css 3
src > components > @ Login.jsx ...
1  import React, { useState } from "react";
2  import api from "../services/api";
3
4  const Login = ({ onLogin }) => {
5    const [username, setUsername] = useState("");
6    const [password, setPassword] = useState("");
7    const [loading, setLoading] = useState(false);
8    const [error, setError] = useState("");
9
10   const handleSubmit = async (e) => {
11     e.preventDefault();
12     setLoading(true);
13     setError("");
14
15     try {
16       const response = await api.post("/auth/login", { username, password });
17       const { token } = response.data;
18       localStorage.setItem("token", token);
19       onLogin();
20     } catch {
21       setError("Invalid credentials. Please try again.");
22     } finally {
23       setLoading(false);
24     }
25   };
26
27   return (
28     <div className="min-h-screen flex">
29      /* Left illustration / gradient */
30      <div className="hidden lg:flex flex-col justify-center items-center w-1/2 bg-gradient-to-br from-[#004d7a] via-[#008793] to-[#00bf72] text-white p-12">
31        <div className="text-center space-y-4">
32          <div className="text-5xl font-bold">JWT Auth</div>
33          <p className="text-lg opacity-90 max-w-sm">
34            Securely access your dashboard and manage your session in real-time.
35          </p>
36        </div>
37        <div className="mt-10">
38          <svg
39            xmlns="http://www.w3.org/2000/svg"
40            className="w-64 h-64 opacity-90"
41            viewBox="0 0 24 24"
42            fill="none"
43            stroke="white"
44            strokeWidth="1.5"
45          >
46            <path d="M12 22c5.523 0 10-4.477 10-10S17.523 2 12 2 2 6.477 2 12S4.477 10 10 10z" />
47            <path d="M9 12l2 2 4-4" strokeLinecap="round" strokeLinejoin="round" />

```

```
# App.css      Login.jsx  Dashboard.jsx  # index.css 3
src > components > @ Loginjsx > ...
4  const Login = ({ onLogin }) => {
48    </svg>
49    </div>
50  </div>
51
52  /* Right Login Section */
53  <div className="flex flex-col justify-center items-center w-full lg:w-1/2 bg-[#f8fafc]">
54    <div className="w-full max-w-sm bg-white/60 backdrop-blur-xl rounded-2xl shadow-xl p-8 border border-gray-200">
55      <div className="mx-auto w-14 h-14 bg-gradient-to-r from-cyan-500 to-blue-600 rounded-xl flex items-center justify-center shadow-md mb-4">
56        <svg>
57          <className="w-8 h-8 text-white"
58            fill="none"
59            stroke="currentColor"
60            viewBox="0 0 24 24"
61          >
62            <path
63              strokeLinecap="round"
64              strokeLinejoin="round"
65              strokeWidth={2}
66              d="M12 15v2m-6 4h12a2 2 0 00-2v-6a2 2 0 00-2-2H6a2 2 0 00-2 2v6a2 2 0 002 2zm10-10V7a4 4 0 00-8 0v4h8z"
67            />
68        </svg>
69      </div>
70      <h2 className="text-2xl font-bold text-gray-800 mb-1">
71        Welcome Back
72      </h2>
73      <p className="text-gray-500 text-sm">
74        Sign in to continue to your dashboard
75      </p>
76    </div>
77
78    <form className="space-y-5" onSubmit={onSubmit}>
79      <div>
80        <label
81          htmlFor="username"
82          className="block text-sm font-medium text-gray-700 mb-1"
83        >
84          Username
85        </label>
86        <input
87          id="username"
88          type="text"
89          className="w-full px-4 py-3 border border-gray-300 rounded-xl focus:ring-2 focus:ring-blue-500 focus:border-transparent"
90          placeholder="Enter your username"
91          value={username}
92          onChange={(e) => setUsername(e.target.value)}
93        />
94      </div>
95
96      <div>
97        <label
98          htmlFor="password"
99          className="block text-sm font-medium text-gray-700 mb-1"
100        >
101          Password
102        </label>
103        <input
104          id="password"
105          type="password"
106          className="w-full px-4 py-3 border border-gray-300 rounded-xl focus:ring-2 focus:ring-blue-500 focus:border-transparent"
107          placeholder="Enter your password"
108          value={password}
109          onChange={(e) => setPassword(e.target.value)}
110        />
111      </div>
112
113      {error &&
114       <div className="text-red-500 text-center text-sm">{error}</div>
115     }
116
117      <button
118        type="submit"
119        className="w-full py-3 bg-gradient-to-r from-cyan-500 to-blue-600 text-white font-semibold rounded-xl shadow-md hover:from-cyan-400 hover:to-blue-500 transition"
120      >
121        {loading ? "Signing in..." : "Sign In"}
122      </button>
123    </form>
124
125    <p className="text-xs text-gray-500 text-center mt-6">
126      Use: <span className="font-medium text-blue-600">student</span> /(< " ">
127      <span className="font-medium text-blue-600">password123</span>
128      </p>
129    </div>
130  </div>
131
132  );
133};
134
135
136 export default Login;
137
```

```
# App.css      Login.jsx  Dashboard.jsx  # index.css 3
src > components > @ Loginjsx > ...
4  const Login = ({ onLogin }) => {
48    </div>
49
50  <div>
51    <div>
52      <label
53        htmlFor="username"
54        className="block text-sm font-medium text-gray-700 mb-1"
55      >
56        Username
57      </label>
58      <input
59        id="username"
60        type="text"
61        className="w-full px-4 py-3 border border-gray-300 rounded-xl focus:ring-2 focus:ring-blue-500 focus:border-transparent"
62        placeholder="Enter your username"
63        value={username}
64        onChange={(e) => setUsername(e.target.value)}
65      />
66    </div>
67
68    <div>
69      <label
70        htmlFor="password"
71        className="block text-sm font-medium text-gray-700 mb-1"
72      >
73        Password
74      </label>
75      <input
76        id="password"
77        type="password"
78        className="w-full px-4 py-3 border border-gray-300 rounded-xl focus:ring-2 focus:ring-blue-500 focus:border-transparent"
79        placeholder="Enter your password"
80        value={password}
81        onChange={(e) => setPassword(e.target.value)}
82      />
83    </div>
84
85    {error &&
86     <div className="text-red-500 text-center text-sm">{error}</div>
87   }
88
89   <button
90     type="submit"
91     className="w-full py-3 bg-gradient-to-r from-cyan-500 to-blue-600 text-white font-semibold rounded-xl shadow-md hover:from-cyan-400 hover:to-blue-500 transition"
92   >
93     {loading ? "Signing in..." : "Sign In"}
94   </button>
95 </form>
96
97
98  <p className="text-xs text-gray-500 text-center mt-6">
99    Use: <span className="font-medium text-blue-600">student</span> /(< " ">
100    <span className="font-medium text-blue-600">password123</span>
101    </p>
102  </div>
103
104  );
105
106  export default Login;
107
```

App.css

```
App.css App.jsx Login.jsx Dashboard.jsx # index.css 3
src > # App.css > ...
1  /* Glassmorphism Global Theme */
2  :root {
3    --glass-bg: #rgba(255, 255, 255, 0.1);
4    --glass-border: #rgba(255, 255, 255, 0.25);
5    --glass-blur: 15px;
6    --accent-gradient: linear-gradient(135deg, #6366f1, #a855f7, #ec4899);
7    --text-color: #f8fafc;
8  }
9
10 body {
11   background: radial-gradient(circle at top left, #lelb4b, #111827, #0f172a);
12   background-attachment: fixed;
13   color: var(--text-color);
14   font-family: "Inter", sans-serif;
15   overflow-x: hidden;
16 }
17
18 #root {
19   max-width: 1280px;
20   margin: 0 auto;
21   padding: 2rem;
22 }
23
24 .glass-card {
25   background: var(--glass-bg);
26   border: 1px solid var(--glass-border);
27   backdrop-filter: blur(var(--glass-blur));
28   border-radius: 1.5rem;
29   box-shadow: 0 4px 40px #rgba(0, 0, 0, 0.25);
30   transition: all 0.3s ease-in-out;
31 }
32
33 .glass-card:hover {
34   transform: translateY(-2px);
35   box-shadow: 0 8px 50px #rgba(0, 0, 0, 0.35);
36 }
37
38 .glass-button {
39   background: var(--accent-gradient);
40   border: none;
41   color: #white;
42   font-weight: 600;
43   border-radius: 1rem;
44   padding: 0.75rem 1.5rem;
45   transition: all 0.3s ease-in-out;
46 }
47
48 .glass-button:hover {
```

```
src > # App.css > ...
49 .glass-card {
50   background: var(--glass-bg);
51   border: 1px solid var(--glass-border);
52   backdrop-filter: blur(var(--glass-blur));
53   border-radius: 1.5rem;
54   box-shadow: 0 4px 40px #rgba(0, 0, 0, 0.25);
55   transition: all 0.3s ease-in-out;
56 }
57
58 .glass-card:hover {
59   transform: translateY(-2px);
60   box-shadow: 0 8px 50px #rgba(0, 0, 0, 0.35);
61 }
62
63 .glass-button {
64   background: var(--accent-gradient);
65   border: none;
66   color: #white;
67   font-weight: 600;
68   border-radius: 1rem;
69   padding: 0.75rem 1.5rem;
70   transition: all 0.3s ease-in-out;
71 }
72
73 .glass-button:hover {
74   filter: brightness(1.2);
75   transform: scale(1.03);
76 }
77
78 input.glass-input {
79   background: #rgba(255, 255, 255, 0.05);
80   border: 1px solid #rgba(255, 255, 255, 0.3);
81   border-radius: 0.75rem;
82   color: #white;
83   padding: 0.75rem 1rem;
84   outline: none;
85   width: 100%;
86   transition: all 0.3s;
87 }
88
89 input.glass-input:focus {
90   border-color: #a855f7;
91   box-shadow: 0 0 8px #rgba(168, 85, 247, 0.5);
92 }
```

Output:

The image displays three views of a JWT Auth application:

- Login Screen:** A teal-colored page titled "JWT Auth" with the subtext "Securely access your dashboard and manage your session in real-time." It features a large white circle containing a checkmark icon.
- Welcome Back Screen:** A white card titled "Welcome Back" with a lock icon. It says "Sign in to continue to your dashboard". It includes fields for "Username" (student) and "Password" (redacted), and a "Sign In" button. Below the button is the text "Use: student / password123".
- Dashboard Screen:** A dark blue dashboard for a "Student User". It shows:
 - Profile:** "Authenticated successfully!" and "Email: student@example.com".
 - Session Timer:** "59:30" (Token expiration countdown).
 - Quick Actions:** Buttons for "View Reports" and "Edit Profile".The footer of the dashboard contains the text "© 2025 JWT Auth App — Built with ❤️ using React + Spring Boot".