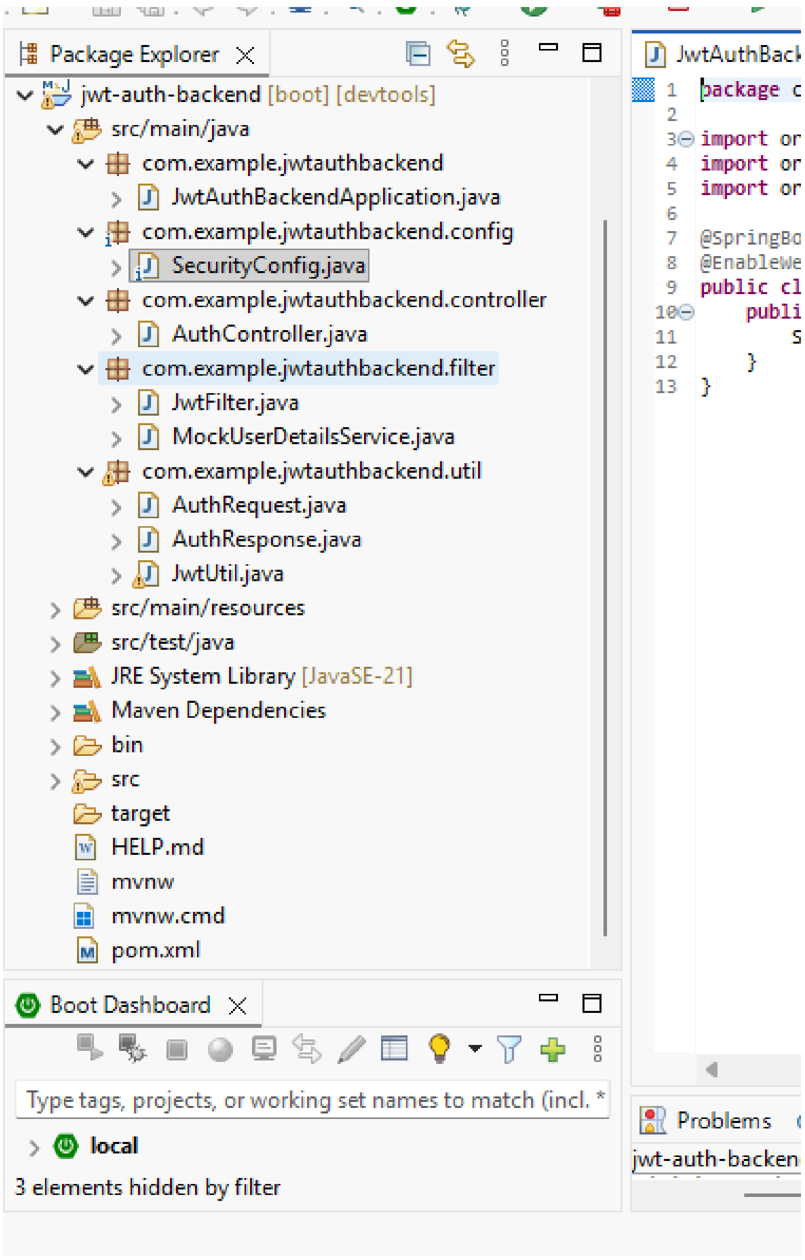
Daily Assignment: 6-11/ November/2025

Backend:



```
jwt-auth-backend [boot] [devtools]
  src/main/java
    com.example.jwtauthbackend
      JwtAuthBackendApplication.java
    com.example.jwtauthbackend.config
      SecurityConfig.java
    com.example.jwtauthbackend.controller
      AuthController.java
    com.example.jwtauthbackend.filter
      JwtFilter.java
      MockUserDetailsService.java
    com.example.jwtauthbackend.util
      AuthRequest.java
      AuthResponse.java
      JwtUtil.java
  src/main/resources
  src/test/java
  JRE System Library [JavaSE-21]
  Maven Dependencies
  bin
  src
  target
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
```

```
package c
import or
import or
import or
@SpringBo
@EnableWe
public cl
    publi
        S
    }
}
```

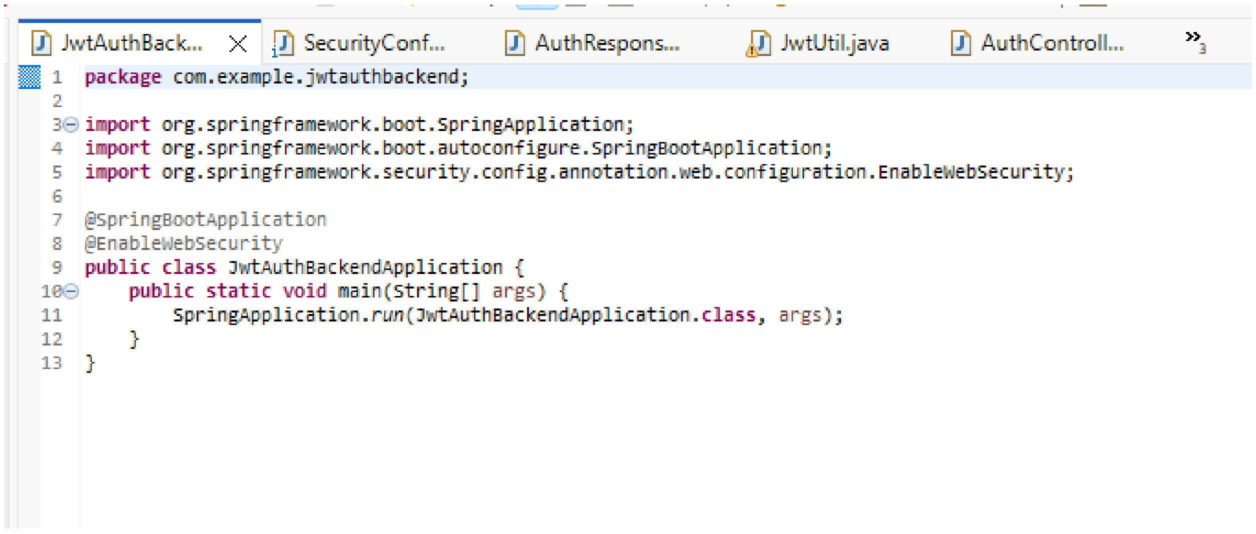
Type tags, projects, or working set names to match (incl. *

local

3 elements hidden by filter

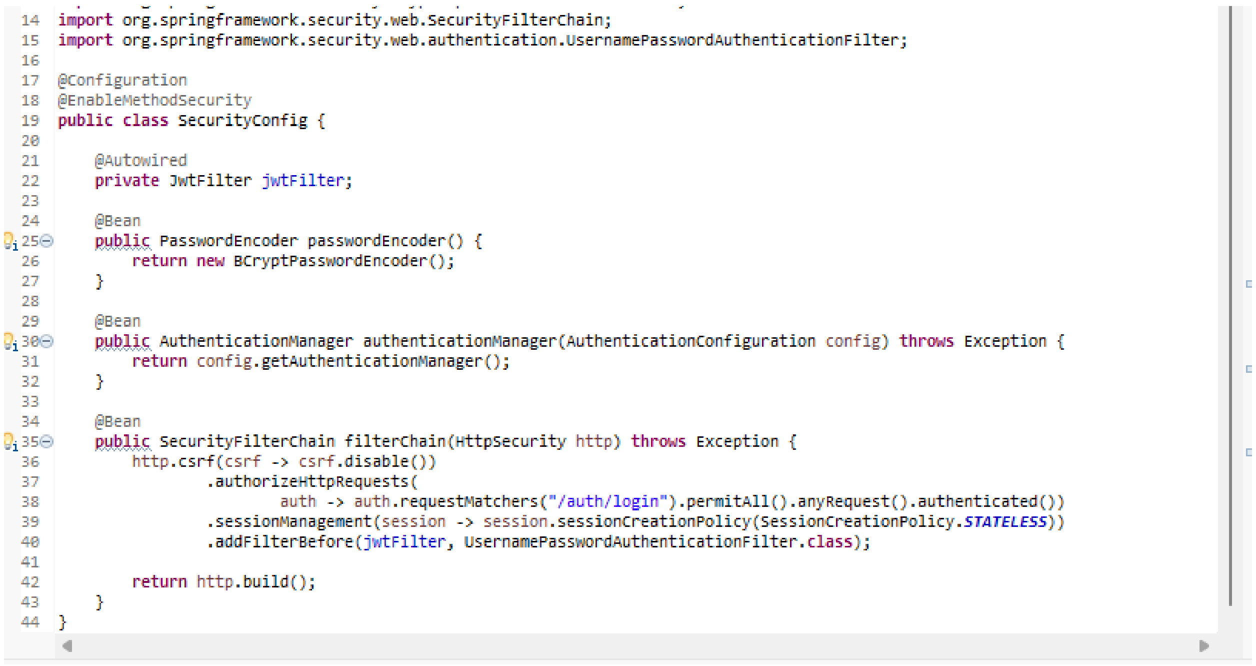
jwt-auth-backen

Jwt auth backend application.java:



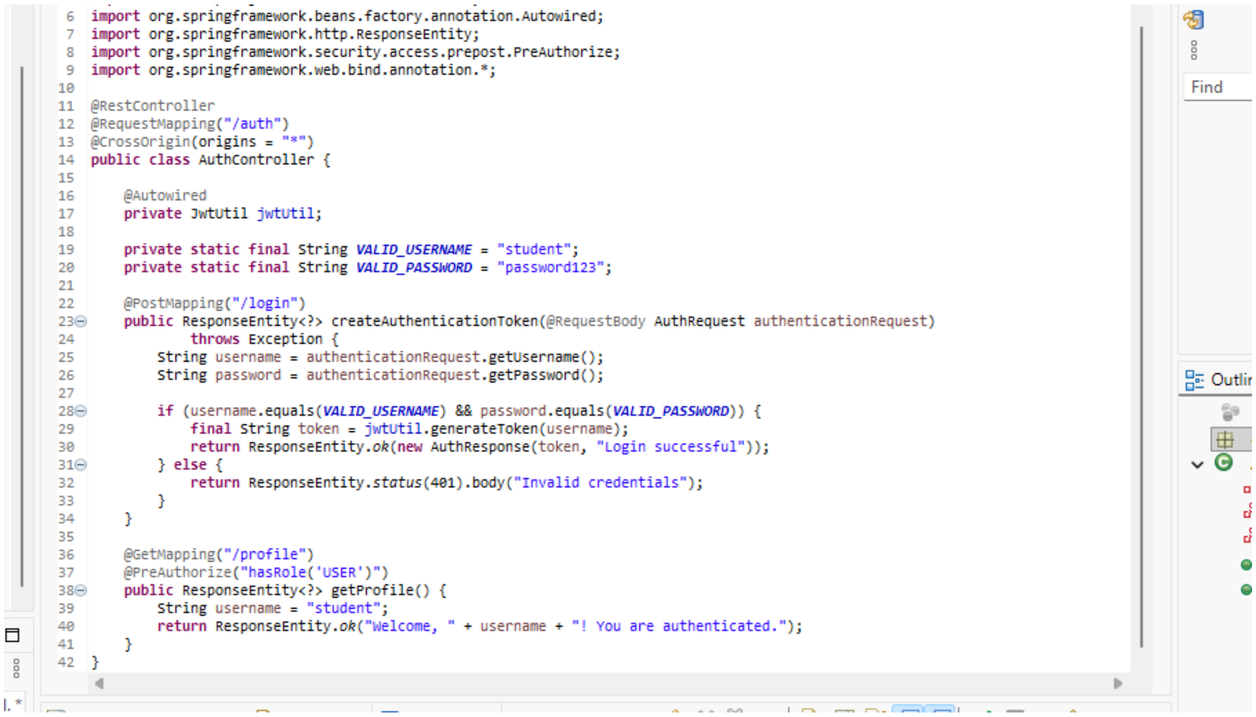
```java
package com.example.jwtauthbackend;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

@SpringBootApplication
@EnableWebSecurity
public class JwtAuthBackendApplication {
    public static void main(String[] args) {
        SpringApplication.run(JwtAuthBackendApplication.class, args);
    }
}
```

Security config.java



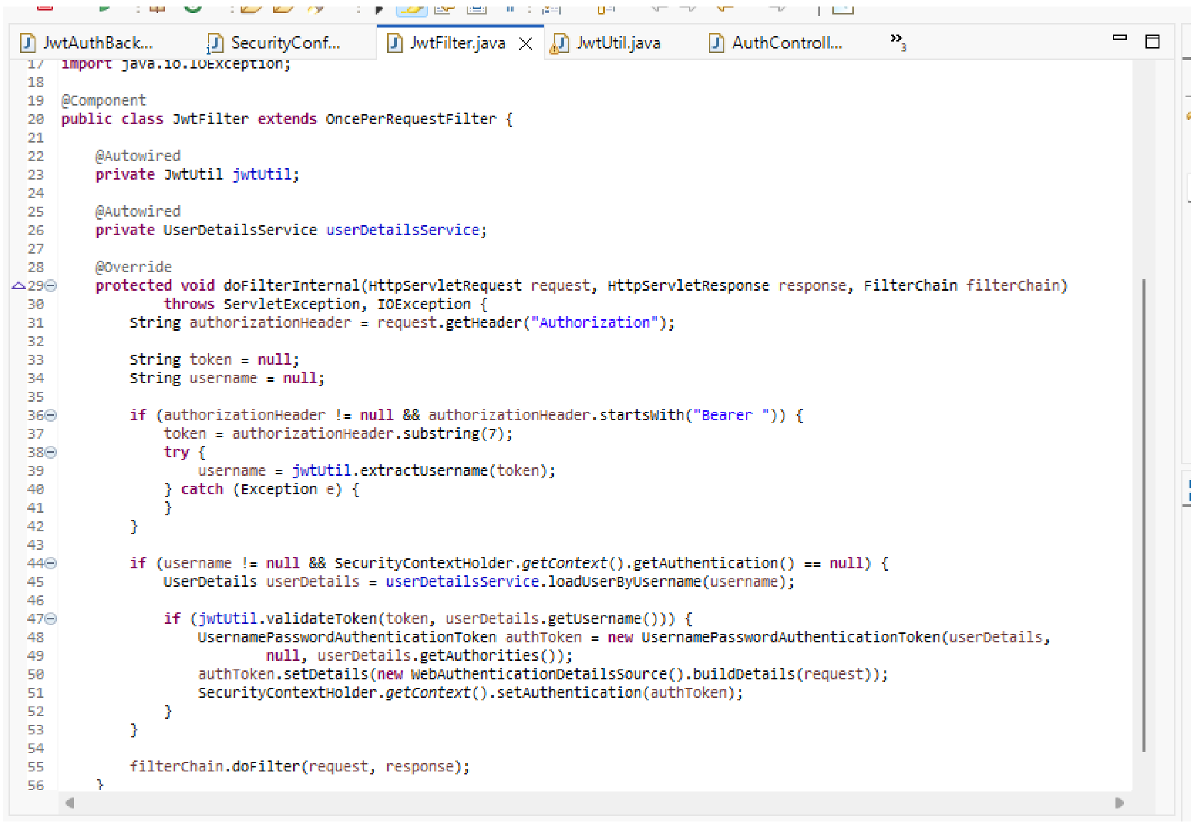
```java
import org.springframework.security.web.SecurityFilterChain;
import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

@Configuration
@EnableMethodSecurity
public class SecurityConfig {

    @Autowired
    private JwtFilter jwtFilter;

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }

    @Bean
    public AuthenticationManager authenticationManager(AuthenticationConfiguration config) throws Exception {
        return config.getAuthenticationManager();
    }

    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http.csrf(csrf -> csrf.disable())
                .authorizeHttpRequests(
                        auth -> auth.requestMatchers("/auth/login").permitAll().anyRequest().authenticated())
                .sessionManagement(session -> session.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
                .addFilterBefore(jwtFilter, UsernamePasswordAuthenticationFilter.class);

        return http.build();
    }
}
```

Authcontroller.java



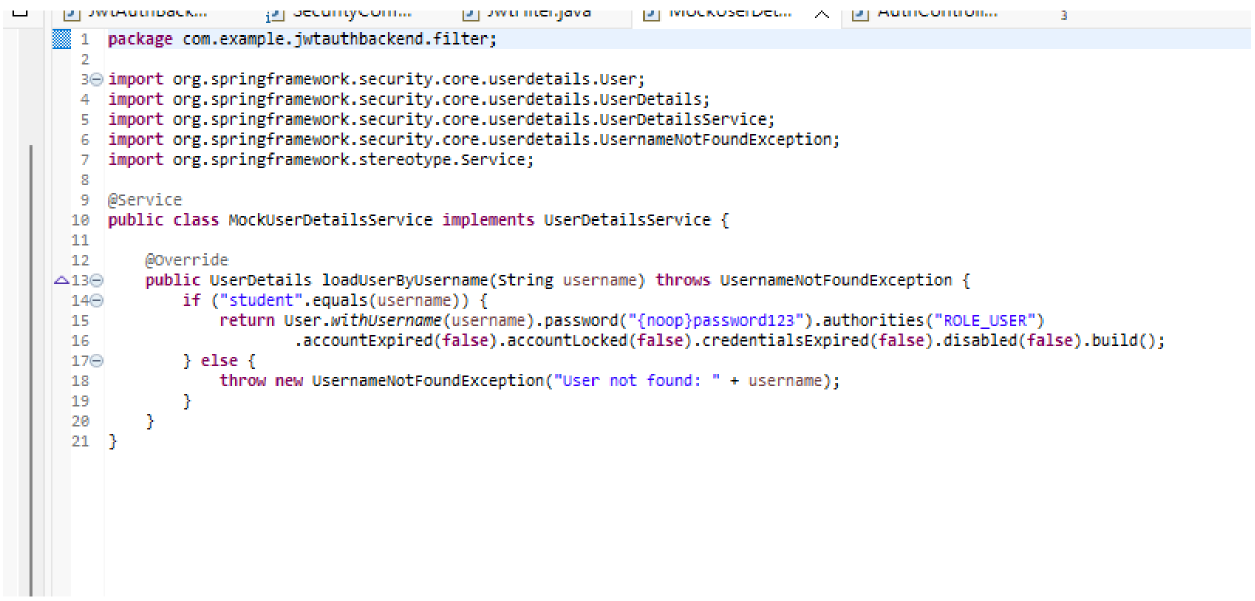
```java
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/auth")
@CrossOrigin(origins = "*")
public class AuthController {

    @Autowired
    private JwtUtil jwtUtil;

    private static final String VALID_USERNAME = "student";
    private static final String VALID_PASSWORD = "password123";

    @PostMapping("/login")
    public ResponseEntity<?> createAuthenticationToken(@RequestBody AuthRequest authenticationRequest)
        throws Exception {
        String username = authenticationRequest.getUsername();
        String password = authenticationRequest.getPassword();

        if (username.equals(VALID_USERNAME) && password.equals(VALID_PASSWORD)) {
            final String token = jwtUtil.generateToken(username);
            return ResponseEntity.ok(new AuthResponse(token, "Login successful"));
        } else {
            return ResponseEntity.status(401).body("Invalid credentials");
        }
    }

    @GetMapping("/profile")
    @PreAuthorize("hasRole('USER')")
    public ResponseEntity<?> getProfile() {
        String username = "student";
        return ResponseEntity.ok("Welcome, " + username + "! You are authenticated.");
    }
}
```

Jwtfilter.java:



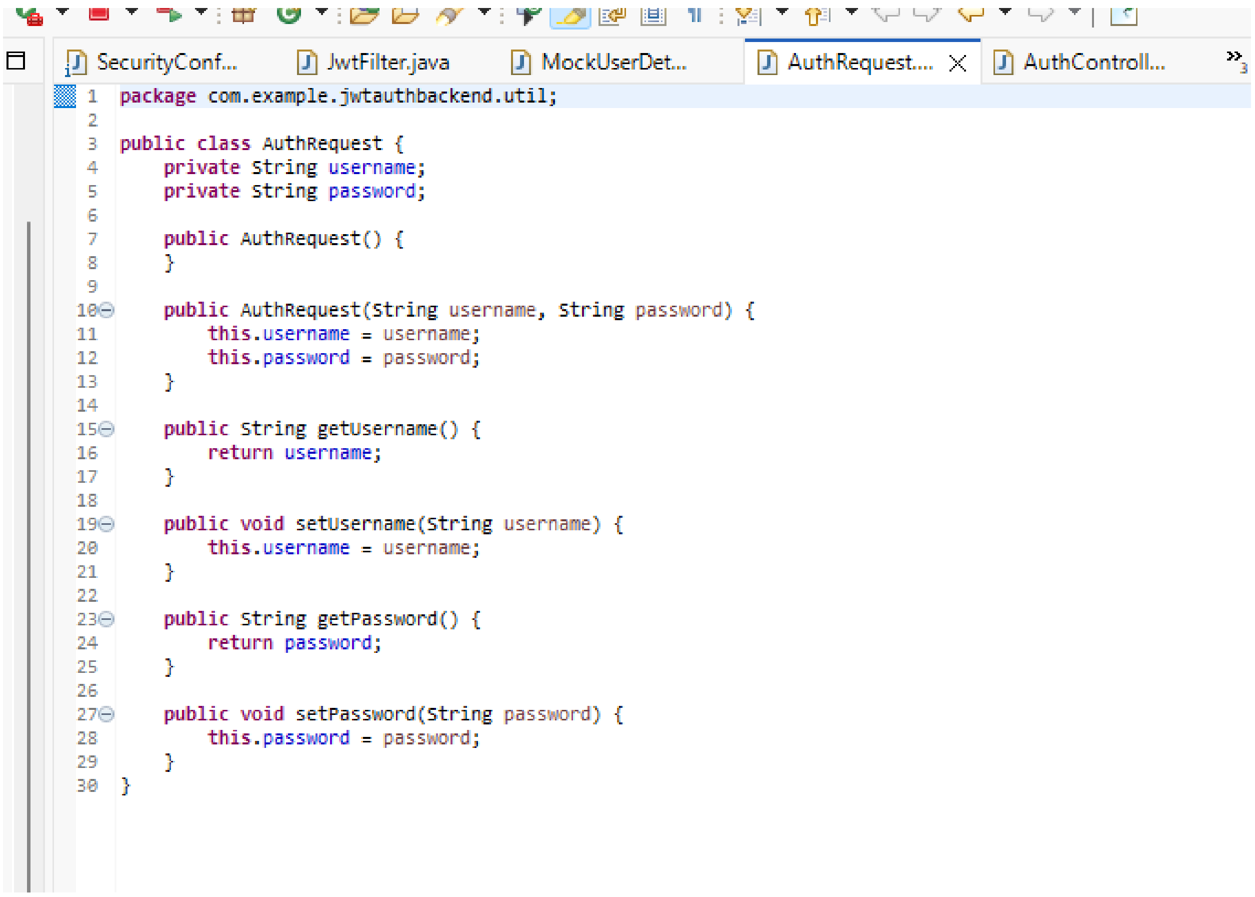
```java
import java.io.IOException;

@Component
public class JwtFilter extends OncePerRequestFilter {

    @Autowired
    private JwtUtil jwtUtil;

    @Autowired
    private UserDetailsService userDetailsService;

    @Override
    protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain)
            throws ServletException, IOException {
        String authorizationHeader = request.getHeader("Authorization");

        String token = null;
        String username = null;

        if (authorizationHeader != null && authorizationHeader.startsWith("Bearer ")) {
            token = authorizationHeader.substring(7);
            try {
                username = jwtUtil.extractUsername(token);
            } catch (Exception e) {
            }
        }

        if (username != null && SecurityContextHolder.getContext().getAuthentication() == null) {
            UserDetails userDetails = userDetailsService.loadUserByUsername(username);

            if (jwtUtil.validateToken(token, userDetails.getUsername())) {
                UsernamePasswordAuthenticationToken authToken = new UsernamePasswordAuthenticationToken(userDetails,
                        null, userDetails.getAuthorities());
                authToken.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));
                SecurityContextHolder.getContext().setAuthentication(authToken);
            }
        }

        filterChain.doFilter(request, response);
    }
```

MockuserDetailsServices.java



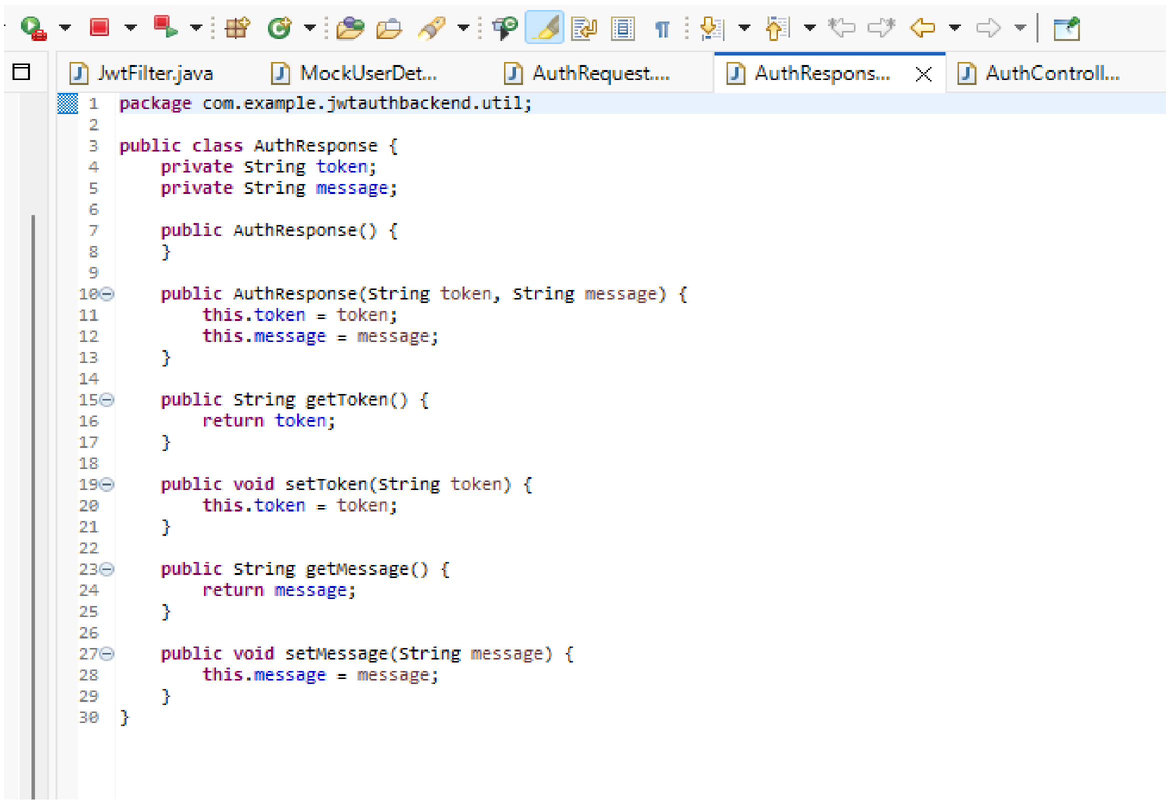
```java
package com.example.jwtauthbackend.filter;

import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

@Service
public class MockUserDetailsService implements UserDetailsService {

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        if ("student".equals(username)) {
            return User.withUsername(username).password("{noop}password123").authorities("ROLE_USER")
                    .accountExpired(false).accountLocked(false).credentialsExpired(false).disabled(false).build();
        } else {
            throw new UsernameNotFoundException("User not found: " + username);
        }
    }
}
```

Authrequest.java



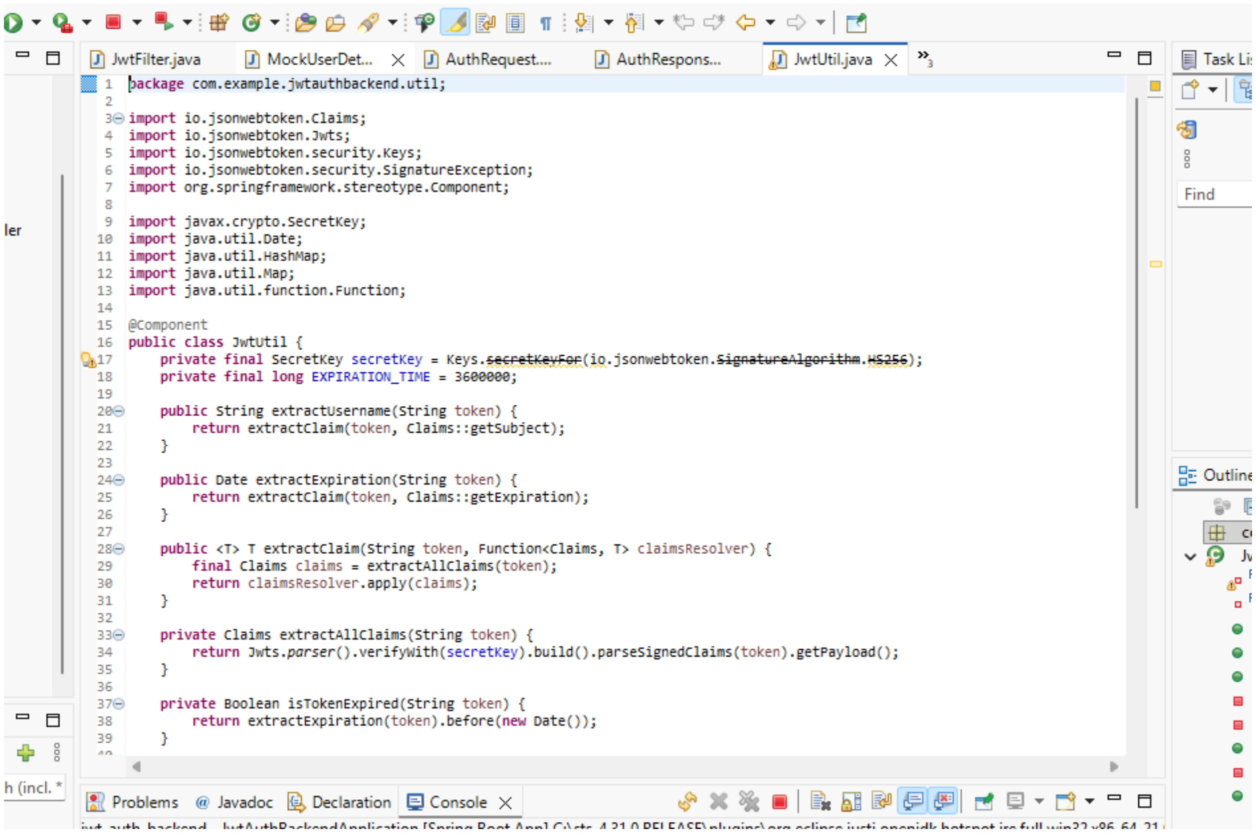
```java
package com.example.jwtauthbackend.util;

public class AuthRequest {
    private String username;
    private String password;

    public AuthRequest() {
    }

    public AuthRequest(String username, String password) {
        this.username = username;
        this.password = password;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

Authresponse.java

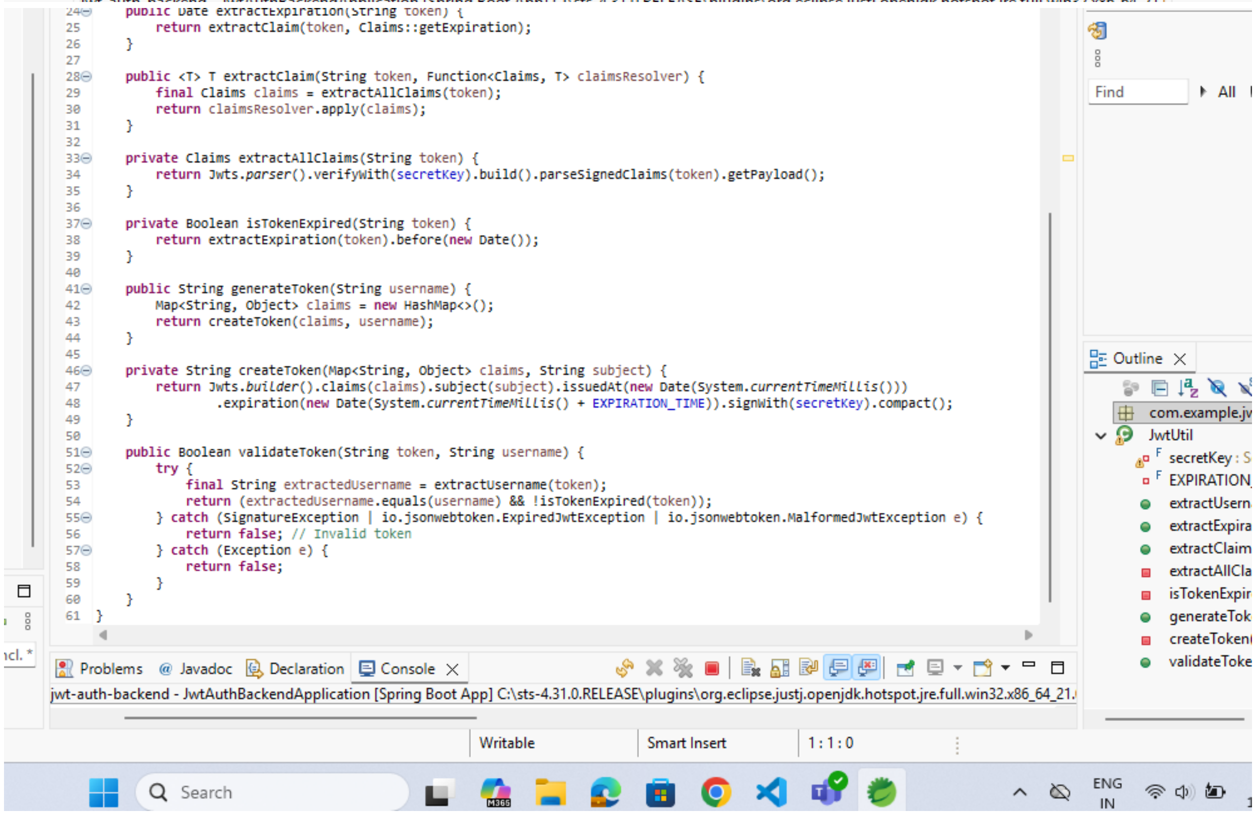


```java
package com.example.jwtauthbackend.util;

public class AuthResponse {
    private String token;
    private String message;

    public AuthResponse() {
    }

    public AuthResponse(String token, String message) {
        this.token = token;
        this.message = message;
    }

    public String getToken() {
        return token;
    }

    public void setToken(String token) {
        this.token = token;
    }

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }
}
```

Jwtutil.java



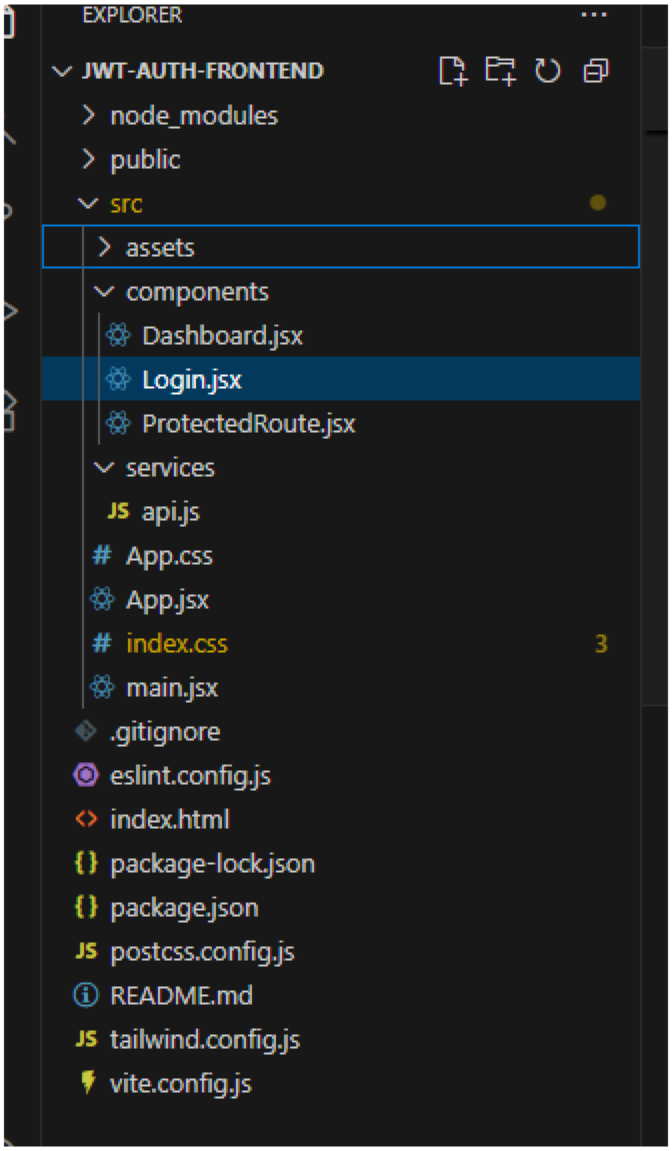
```java
package com.example.jwtauthbackend.util;

import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.security.Keys;
import io.jsonwebtoken.security.SignatureException;
import org.springframework.stereotype.Component;

import javax.crypto.SecretKey;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;
import java.util.function.Function;

@Component
public class JwtUtil {
    private final SecretKey secretKey = Keys.~~secretKeyFor~~(io.jsonwebtoken.~~SignatureAlgorithm~~.~~HS256~~);
    private final long EXPIRATION_TIME = 3600000;

    public String extractUsername(String token) {
        return extractClaim(token, Claims::getSubject);
    }

    public Date extractExpiration(String token) {
        return extractClaim(token, Claims::getExpiration);
    }

    public <T> T extractClaim(String token, Function<Claims, T> claimsResolver) {
        final Claims claims = extractAllClaims(token);
        return claimsResolver.apply(claims);
    }

    private Claims extractAllClaims(String token) {
        return Jwts.parser().verifyWith(secretKey).build().parseSignedClaims(token).getPayload();
    }

    private Boolean isTokenExpired(String token) {
        return extractExpiration(token).before(new Date());
    }
```



```java
    public Date extractExpiration(String token) {
        return extractClaim(token, Claims::getExpiration);
    }

    public <T> T extractClaim(String token, Function<Claims, T> claimsResolver) {
        final Claims claims = extractAllClaims(token);
        return claimsResolver.apply(claims);
    }

    private Claims extractAllClaims(String token) {
        return Jwts.parser().verifyWith(secretKey).build().parseSignedClaims(token).getPayload();
    }

    private Boolean isTokenExpired(String token) {
        return extractExpiration(token).before(new Date());
    }

    public String generateToken(String username) {
        Map<String, Object> claims = new HashMap<>();
        return createToken(claims, username);
    }

    private String createToken(Map<String, Object> claims, String subject) {
        return Jwts.builder().claims(claims).subject(subject).issuedAt(new Date(System.currentTimeMillis()))
                .expiration(new Date(System.currentTimeMillis() + EXPIRATION_TIME)).signWith(secretKey).compact();
    }

    public Boolean validateToken(String token, String username) {
        try {
            final String extractedUsername = extractUsername(token);
            return (extractedUsername.equals(username) && !isTokenExpired(token));
        } catch (SignatureException | io.jsonwebtoken.ExpiredJwtException | io.jsonwebtoken.MalformedJwtException e) {
            return false; // Invalid token
        } catch (Exception e) {
            return false;
        }
    }
}
```

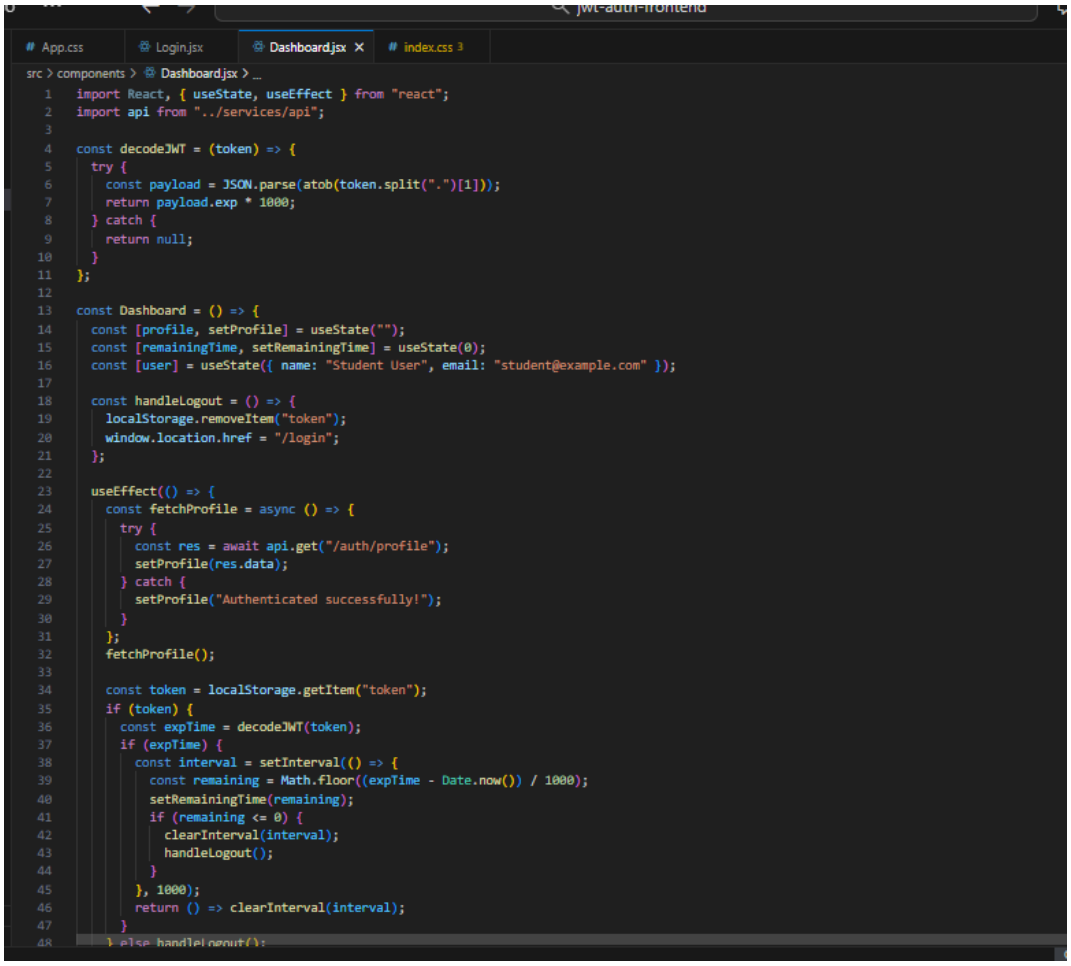
```
jwt-auth-backend - JwtAuthBackendApplication [Spring Boot App] C:\sts-4.31.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.
```

Frontend:

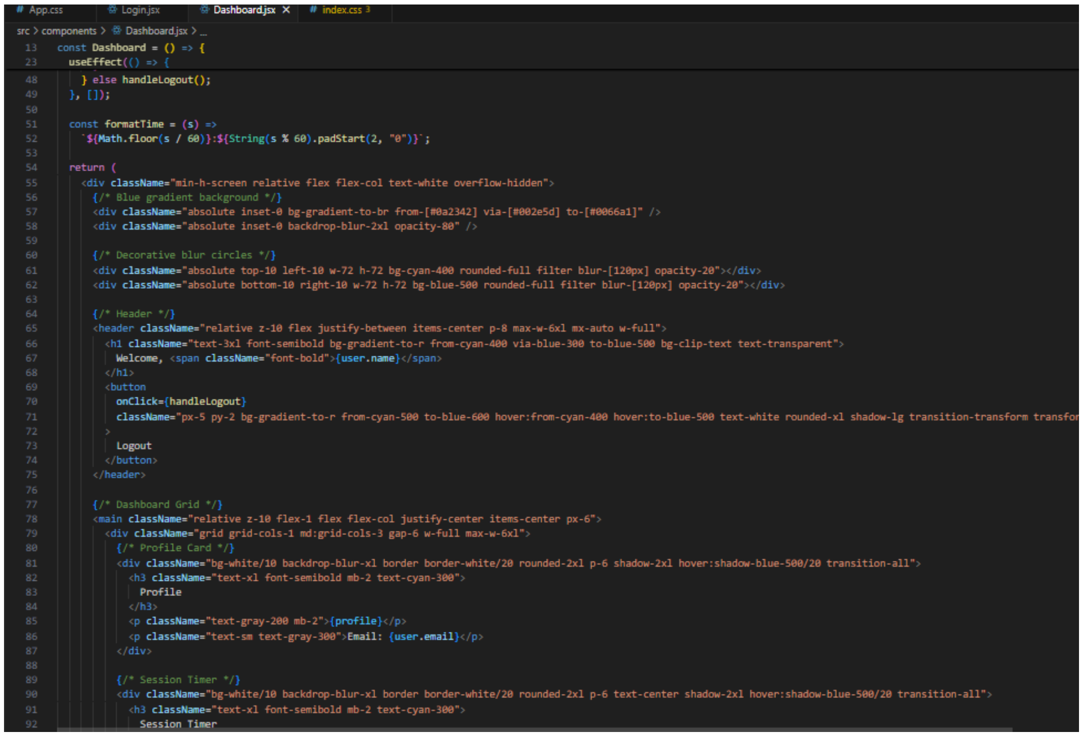


```
EXPLORER
JWT-AUTH-FRONTEND
  node_modules
  public
  src
    assets
    components
      Dashboard.jsx
      Login.jsx
      ProtectedRoute.jsx
    services
      api.js
    App.css
    App.jsx
    index.css                3
    main.jsx
  .gitignore
  eslint.config.js
  index.html
  package-lock.json
  package.json
  postcss.config.js
  README.md
  tailwind.config.js
  vite.config.js
```

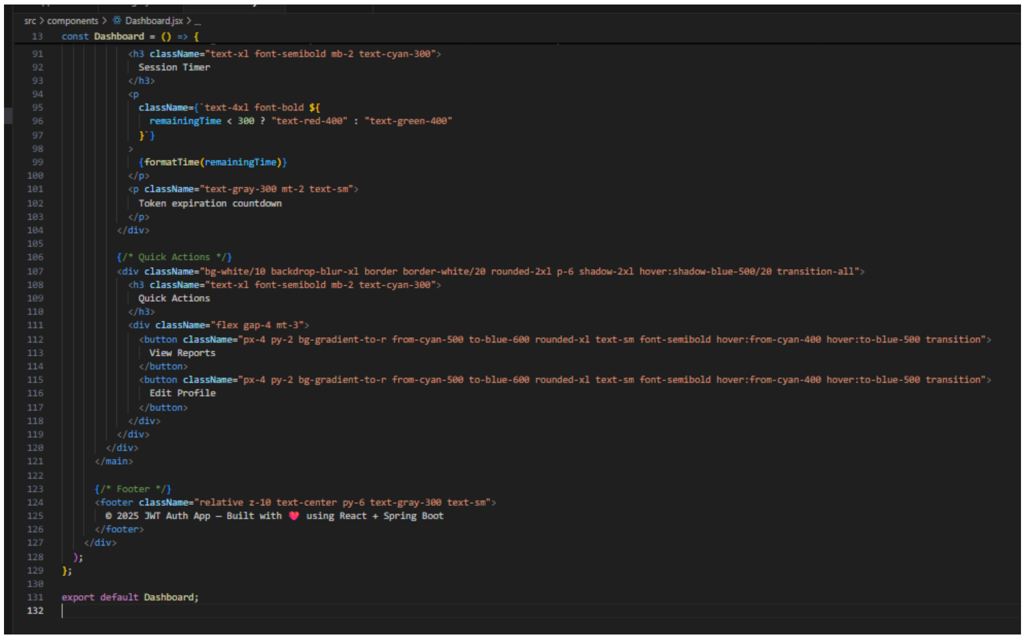
Dashboard.jsx



```jsx
import React, { useState, useEffect } from "react";
import api from "../services/api";

const decodeJWT = (token) => {
  try {
    const payload = JSON.parse(atob(token.split(".")[1]));
    return payload.exp * 1000;
  } catch {
    return null;
  }
};

const Dashboard = () => {
  const [profile, setProfile] = useState("");
  const [remainingTime, setRemainingTime] = useState(0);
  const [user] = useState({ name: "Student User", email: "student@example.com" });

  const handleLogout = () => {
    localStorage.removeItem("token");
    window.location.href = "/login";
  };

  useEffect(() => {
    const fetchProfile = async () => {
      try {
        const res = await api.get("/auth/profile");
        setProfile(res.data);
      } catch {
        setProfile("Authenticated successfully!");
      }
    };
    fetchProfile();

    const token = localStorage.getItem("token");
    if (token) {
      const expTime = decodeJWT(token);
      if (expTime) {
        const interval = setInterval(() => {
          const remaining = Math.floor((expTime - Date.now()) / 1000);
          setRemainingTime(remaining);
          if (remaining <= 0) {
            clearInterval(interval);
            handleLogout();
          }
        }, 1000);
        return () => clearInterval(interval);
      }
    } else handleLogout();
```

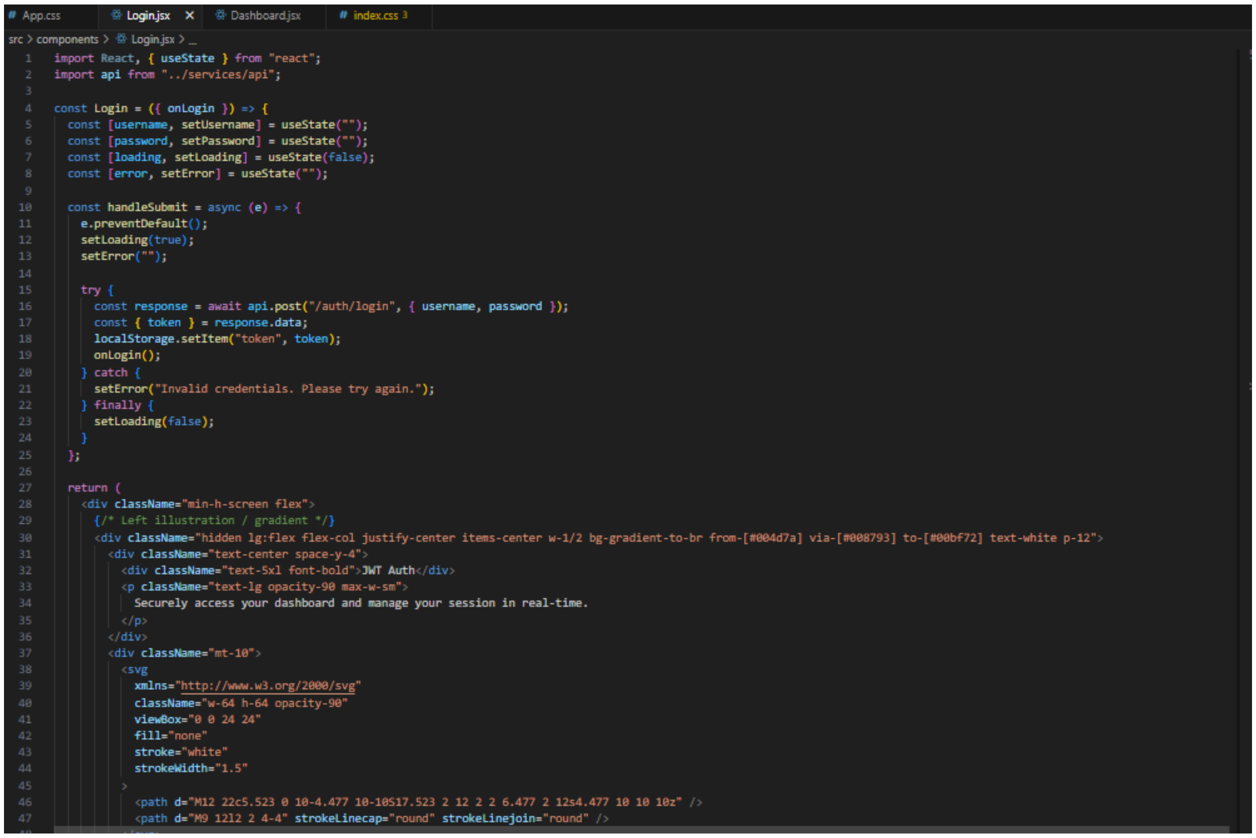


```jsx
const Dashboard = () => {
  useEffect(() => {
    } else handleLogout();
  }, []);

  const formatTime = (s) =>
    `${Math.floor(s / 60)}:${String(s % 60).padStart(2, "0")}`;

  return (
    <div className="min-h-screen relative flex flex-col text-white overflow-hidden">
      {/* Blue gradient background */}
      <div className="absolute inset-0 bg-gradient-to-br from-[#0a2342] via-[#002e5d] to-[#0066a1]" />
      <div className="absolute inset-0 backdrop-blur-2xl opacity-80" />

      {/* Decorative blur circles */}
      <div className="absolute top-10 left-10 w-72 h-72 bg-cyan-400 rounded-full filter blur-[120px] opacity-20"></div>
      <div className="absolute bottom-10 right-10 w-72 h-72 bg-blue-500 rounded-full filter blur-[120px] opacity-20"></div>

      {/* Header */}
      <header className="relative z-10 flex justify-between items-center p-8 max-w-6xl mx-auto w-full">
        <h1 className="text-3xl font-semibold bg-gradient-to-r from-cyan-400 via-blue-300 to-blue-500 bg-clip-text text-transparent">
          Welcome, <span className="font-bold">{user.name}</span>
        </h1>
        <button
          onClick={handleLogout}
          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-xl shadow-lg transition-transform transfor
        >
          Logout
        </button>
      </header>

      {/* Dashboard Grid */}
      <main className="relative z-10 flex-1 flex flex-col justify-center items-center px-6">
        <div className="grid grid-cols-1 md:grid-cols-3 gap-6 w-full max-w-6xl">
          {/* Profile Card */}
          <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">
            <h3 className="text-xl font-semibold mb-2 text-cyan-300">
              Profile
            </h3>
            <p className="text-gray-200 mb-2">{profile}</p>
            <p className="text-sm text-gray-300">Email: {user.email}</p>
          </div>

          {/* Session Timer */}
          <div className="bg-white/10 backdrop-blur-xl border border-white/20 rounded-2xl p-6 text-center shadow-2xl hover:shadow-blue-500/20 transition-all">
            <h3 className="text-xl font-semibold mb-2 text-cyan-300">
              Session Timer
```

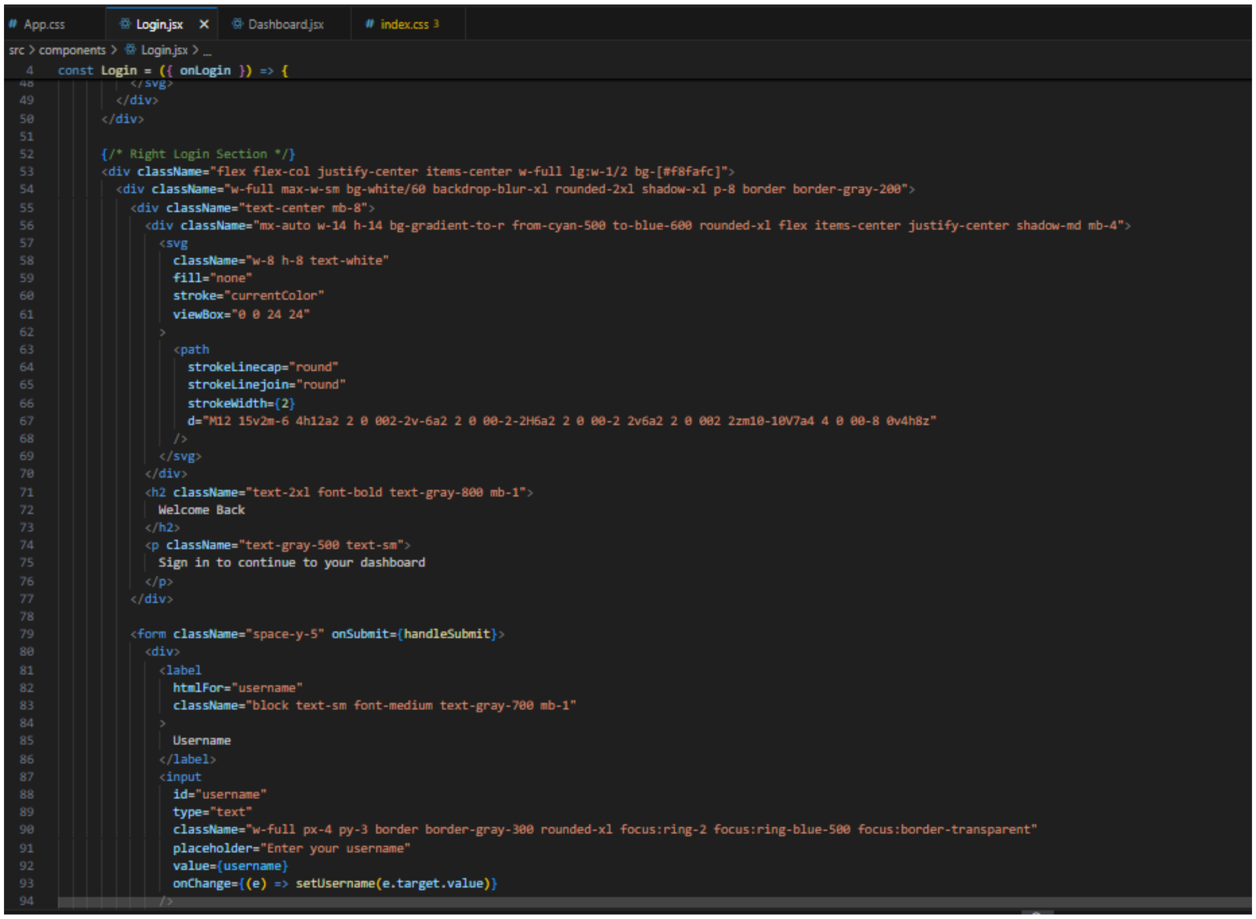


```jsx
const Dashboard = () => {
            <h3 className="text-xl font-semibold mb-2 text-cyan-300">
              Session Timer
            </h3>
            <p
              className={`text-4xl font-bold ${
                remainingTime < 300 ? "text-red-400" : "text-green-400"
              }`}
            >
              {formatTime(remainingTime)}
            </p>
            <p className="text-gray-300 mt-2 text-sm">
              Token expiration countdown
            </p>
          </div>

          {/* Quick Actions */}
          <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">
            <h3 className="text-xl font-semibold mb-2 text-cyan-300">
              Quick Actions
            </h3>
            <div className="flex gap-4 mt-3">
              <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">
                View Reports
              </button>
              <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">
                Edit Profile
              </button>
            </div>
          </div>
        </div>
      </main>

      {/* Footer */}
      <footer className="relative z-10 text-center py-6 text-gray-300 text-sm">
        © 2025 JWT Auth App — Built with ❤️ using React + Spring Boot
      </footer>
    </div>
  );
};

export default Dashboard;
```

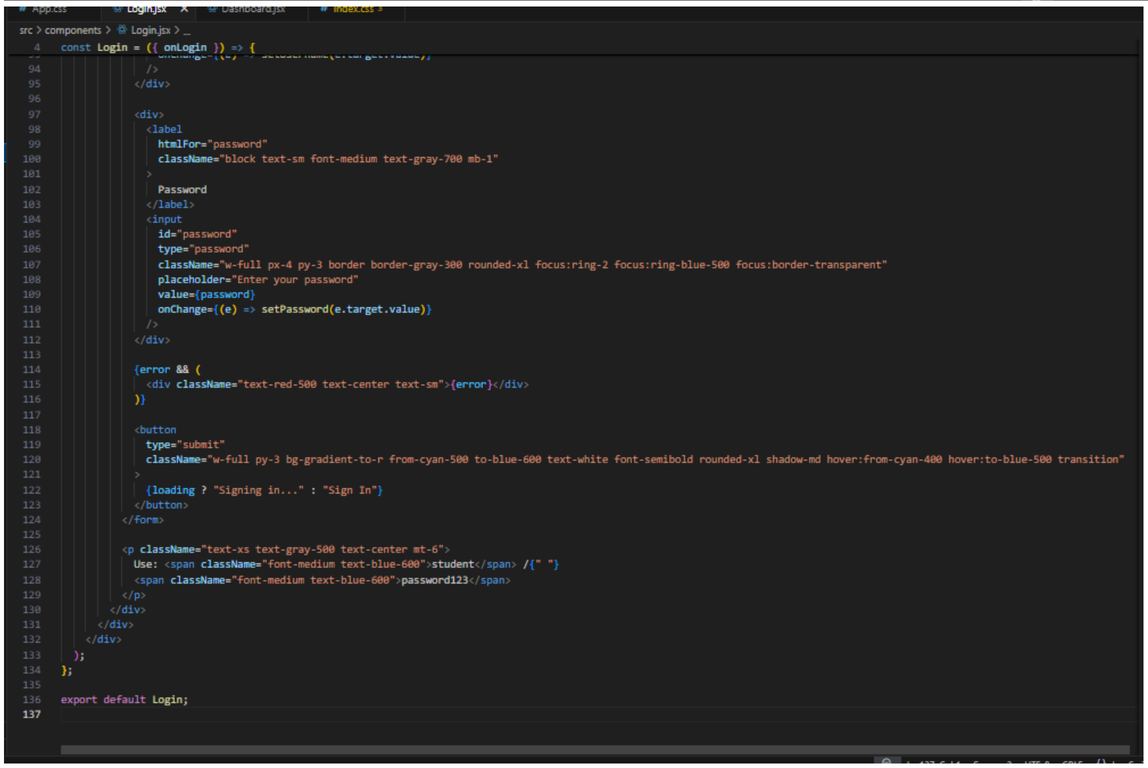
Login.jsx



```jsx
import React, { useState } from "react";
import api from "../services/api";

const Login = ({ onLogin }) => {
  const [username, setUsername] = useState("");
  const [password, setPassword] = useState("");
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState("");

  const handleSubmit = async (e) => {
    e.preventDefault();
    setLoading(true);
    setError("");

    try {
      const response = await api.post("/auth/login", { username, password });
      const { token } = response.data;
      localStorage.setItem("token", token);
      onLogin();
    } catch {
      setError("Invalid credentials. Please try again.");
    } finally {
      setLoading(false);
    }
  };

  return (
    <div className="min-h-screen flex">
      {/* Left illustration / gradient */}
      <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">
        <div className="text-center space-y-4">
          <div className="text-5xl font-bold">JWT Auth</div>
          <p className="text-lg opacity-90 max-w-sm">
            Securely access your dashboard and manage your session in real-time.
          </p>
        </div>
        <div className="mt-10">
          <svg
            xmlns="http://www.w3.org/2000/svg"
            className="w-64 h-64 opacity-90"
            viewBox="0 0 24 24"
            fill="none"
            stroke="white"
            strokeWidth="1.5"
          >
            <path d="M12 22c5.523 0 10-4.477 10-10S17.523 2 12 2 2 6.477 2 12s4.477 10 10 10z" />
            <path d="M9 12l2 2 4-4" strokeLinecap="round" strokeLinejoin="round" />
```

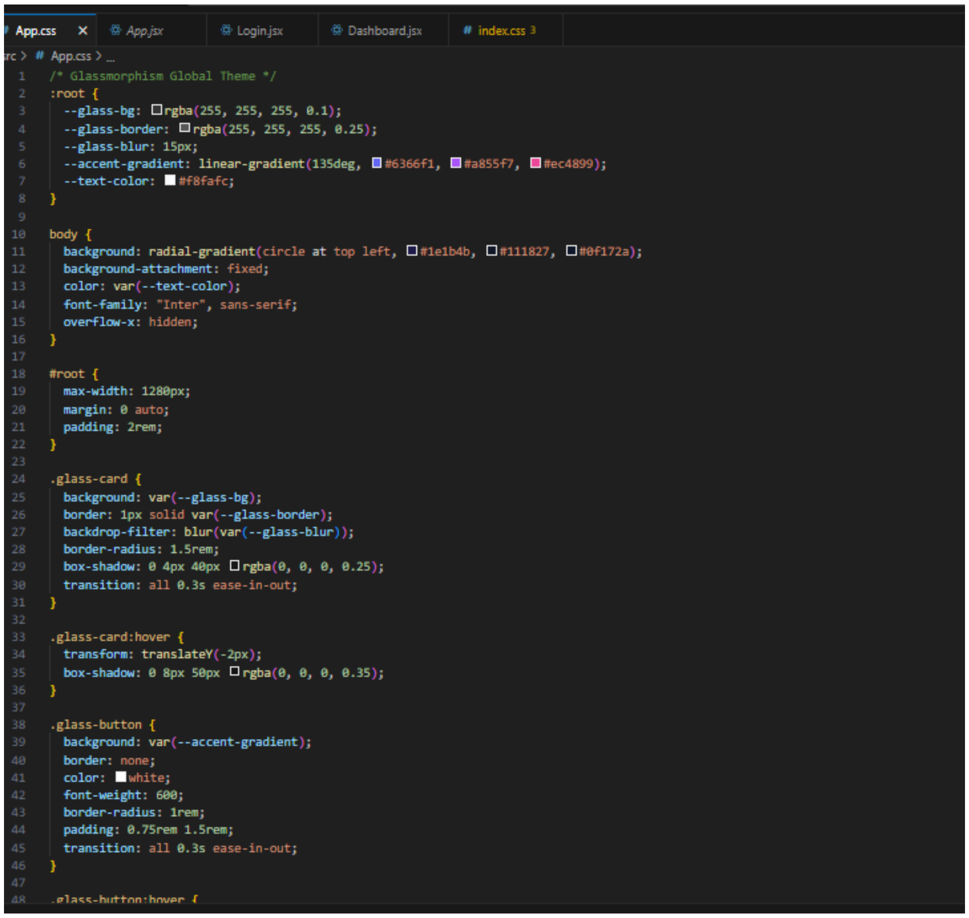


```jsx
const Login = ({ onLogin }) => {
          </svg>
        </div>
      </div>

      {/* Right Login Section */}
      <div className="flex flex-col justify-center items-center w-full lg:w-1/2 bg-[#f8fafc]">
        <div className="w-full max-w-sm bg-white/60 backdrop-blur-xl rounded-2xl shadow-xl p-8 border border-gray-200">
          <div className="text-center mb-8">
            <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">
              <svg
                className="w-8 h-8 text-white"
                fill="none"
                stroke="currentColor"
                viewBox="0 0 24 24"
              >
                <path
                  strokeLinecap="round"
                  strokeLinejoin="round"
                  strokeWidth={2}
                  d="M12 15v2m-6 4h12a2 2 0 002-2v-6a2 2 0 00-2-2H6a2 2 0 00-2 2v6a2 2 0 002 2zm10-10V7a4 4 0 00-8 0v4h8z"
                />
              </svg>
            </div>
            <h2 className="text-2xl font-bold text-gray-800 mb-1">
              Welcome Back
            </h2>
            <p className="text-gray-500 text-sm">
              Sign in to continue to your dashboard
            </p>
          </div>

          <form className="space-y-5" onSubmit={handleSubmit}>
            <div>
              <label
                htmlFor="username"
                className="block text-sm font-medium text-gray-700 mb-1"
              >
                Username
              </label>
              <input
                id="username"
                type="text"
                className="w-full px-4 py-3 border border-gray-300 rounded-xl focus:ring-2 focus:ring-blue-500 focus:border-transparent"
                placeholder="Enter your username"
                value={username}
                onChange={(e) => setUsername(e.target.value)}
              />
```

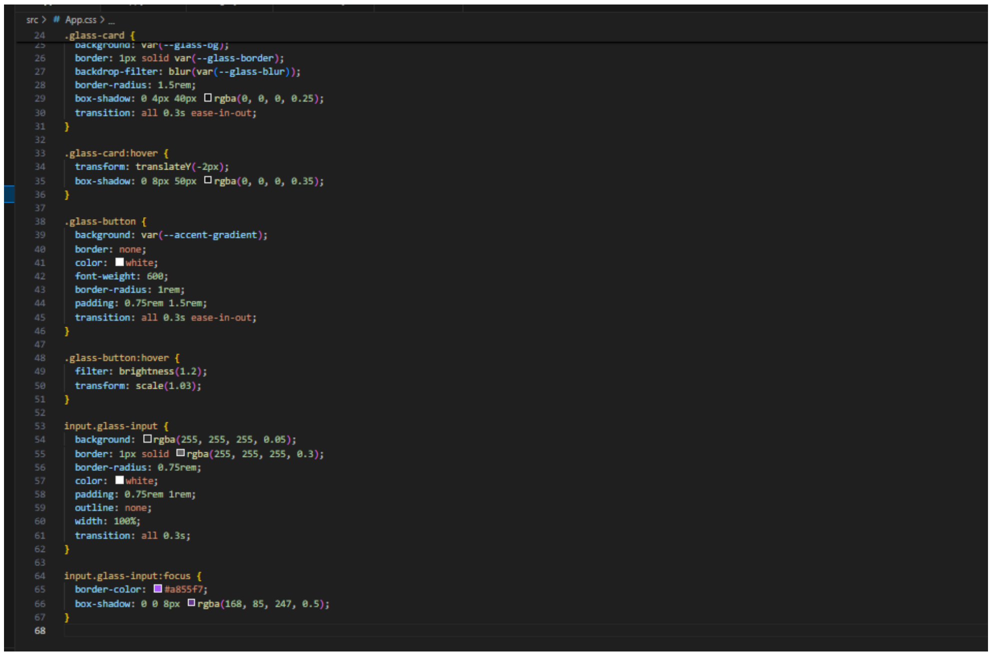


```jsx
const Login = ({ onLogin }) => {
              />
            </div>

            <div>
              <label
                htmlFor="password"
                className="block text-sm font-medium text-gray-700 mb-1"
              >
                Password
              </label>
              <input
                id="password"
                type="password"
                className="w-full px-4 py-3 border border-gray-300 rounded-xl focus:ring-2 focus:ring-blue-500 focus:border-transparent"
                placeholder="Enter your password"
                value={password}
                onChange={(e) => setPassword(e.target.value)}
              />
            </div>

            {error && (
              <div className="text-red-500 text-center text-sm">{error}</div>
            )}

            <button
              type="submit"
              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"
            >
              {loading ? "Signing in..." : "Sign In"}
            </button>
          </form>

          <p className="text-xs text-gray-500 text-center mt-6">
            Use: <span className="font-medium text-blue-600">student</span> /{" "}
            <span className="font-medium text-blue-600">password123</span>
          </p>
        </div>
      </div>
    </div>
  );
};

export default Login;
```

App.css



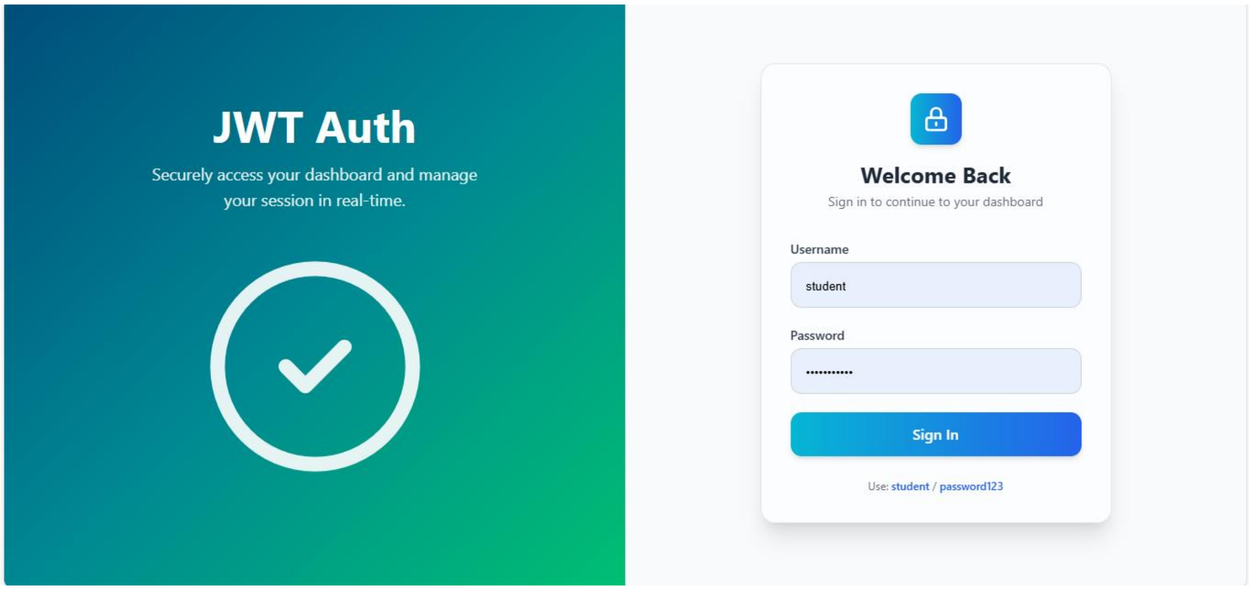
```css
/* Glassmorphism Global Theme */
:root {
  --glass-bg: rgba(255, 255, 255, 0.1);
  --glass-border: rgba(255, 255, 255, 0.25);
  --glass-blur: 15px;
  --accent-gradient: linear-gradient(135deg, #6366f1, #a855f7, #ec4899);
  --text-color: #f8fafc;
}

body {
  background: radial-gradient(circle at top left, #1e1b4b, #111827, #0f172a);
  background-attachment: fixed;
  color: var(--text-color);
  font-family: "Inter", sans-serif;
  overflow-x: hidden;
}

#root {
  max-width: 1280px;
  margin: 0 auto;
  padding: 2rem;
}

.glass-card {
  background: var(--glass-bg);
  border: 1px solid var(--glass-border);
  backdrop-filter: blur(var(--glass-blur));
  border-radius: 1.5rem;
  box-shadow: 0 4px 40px rgba(0, 0, 0, 0.25);
  transition: all 0.3s ease-in-out;
}

.glass-card:hover {
  transform: translateY(-2px);
  box-shadow: 0 8px 50px rgba(0, 0, 0, 0.35);
}

.glass-button {
  background: var(--accent-gradient);
  border: none;
  color: white;
  font-weight: 600;
  border-radius: 1rem;
  padding: 0.75rem 1.5rem;
  transition: all 0.3s ease-in-out;
}

.glass-button:hover {
```



```css
.glass-card {
  background: var(--glass-bg);
  border: 1px solid var(--glass-border);
  backdrop-filter: blur(var(--glass-blur));
  border-radius: 1.5rem;
  box-shadow: 0 4px 40px rgba(0, 0, 0, 0.25);
  transition: all 0.3s ease-in-out;
}

.glass-card:hover {
  transform: translateY(-2px);
  box-shadow: 0 8px 50px rgba(0, 0, 0, 0.35);
}

.glass-button {
  background: var(--accent-gradient);
  border: none;
  color: white;
  font-weight: 600;
  border-radius: 1rem;
  padding: 0.75rem 1.5rem;
  transition: all 0.3s ease-in-out;
}

```

```css
.glass-button:hover {
  filter: brightness(1.2);
  transform: scale(1.03);
}

input.glass-input {
  background: rgba(255, 255, 255, 0.05);
  border: 1px solid rgba(255, 255, 255, 0.3);
  border-radius: 0.75rem;
  color: white;
  padding: 0.75rem 1rem;
  outline: none;
  width: 100%;
  transition: all 0.3s;
}

input.glass-input:focus {
  border-color: #a855f7;
  box-shadow: 0 0 8px rgba(168, 85, 247, 0.5);
}

```

Output:



JWT Auth

Securely access your dashboard and manage your session in real-time.

Welcome Back

Sign in to continue to your dashboard

Username

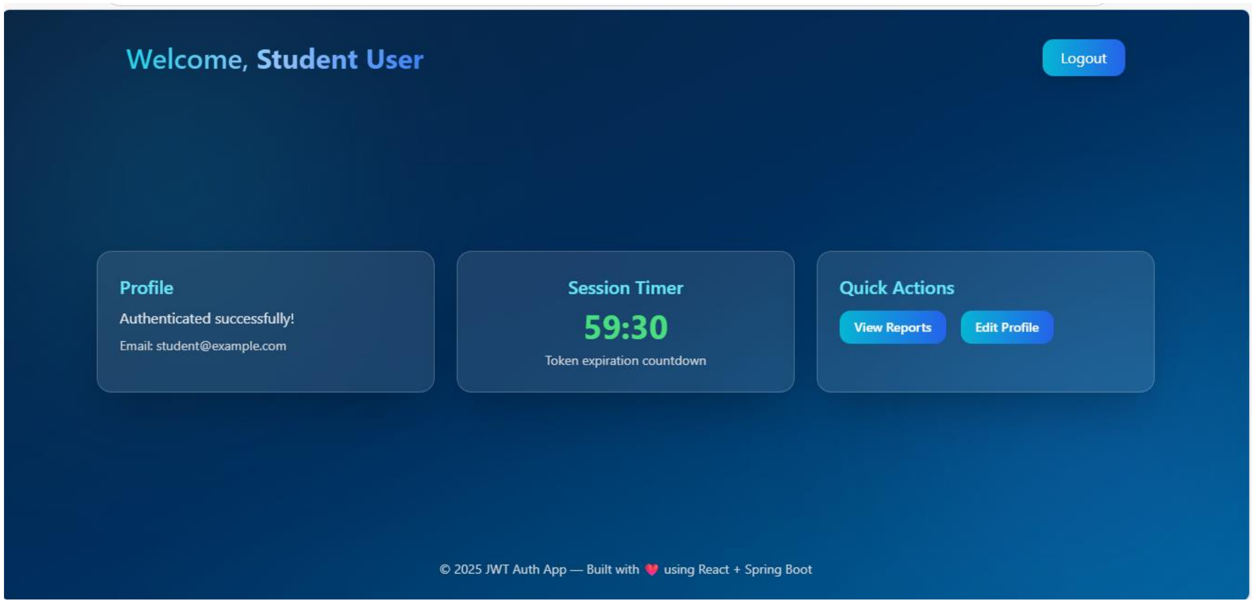
student

Password

••••••••••••

Sign In

Use: student / password123



Welcome, Student User

Logout

Profile

Authenticated successfully!

Email: student@example.com

Session Timer

59:30

Token expiration countdown

Quick Actions

View Reports

Edit Profile

© 2025 JWT Auth App — Built with ❤️ using React + Spring Boot