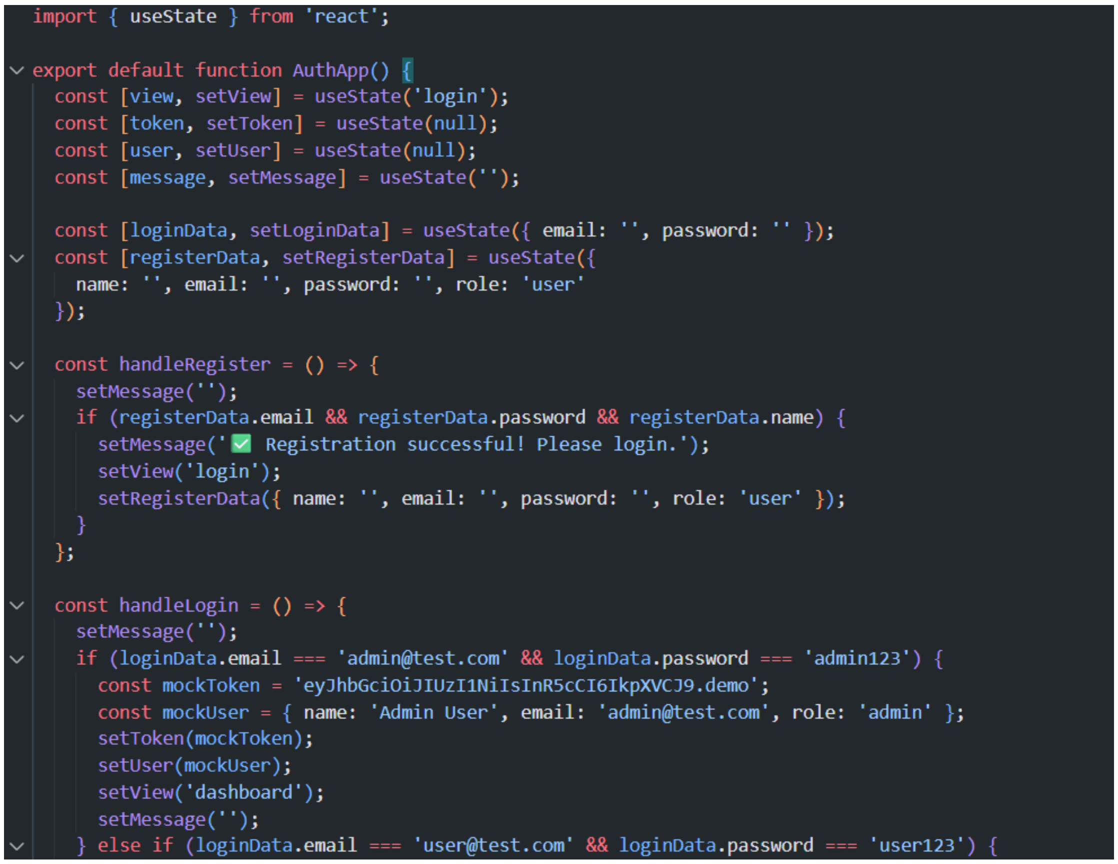
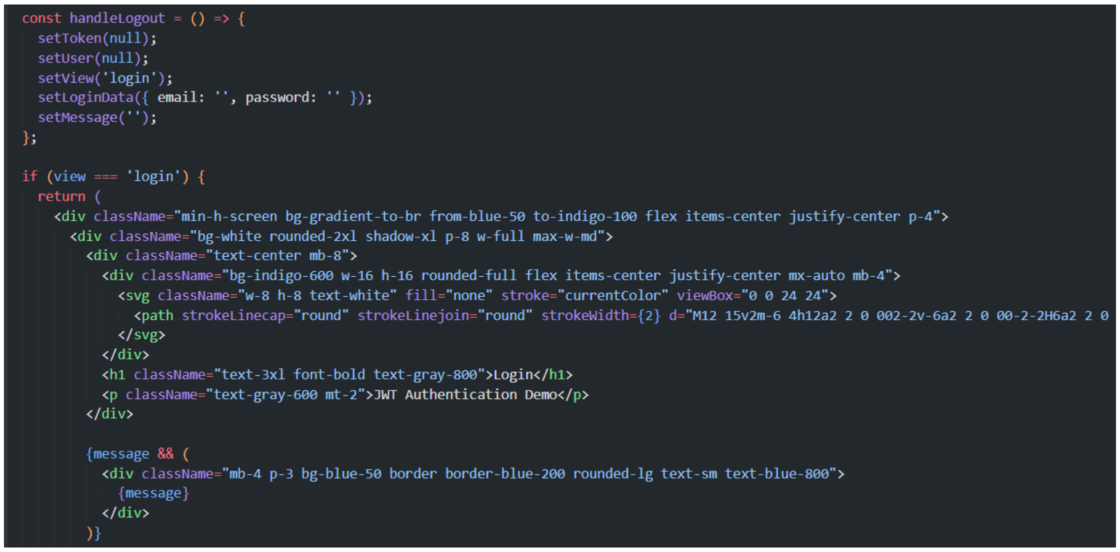
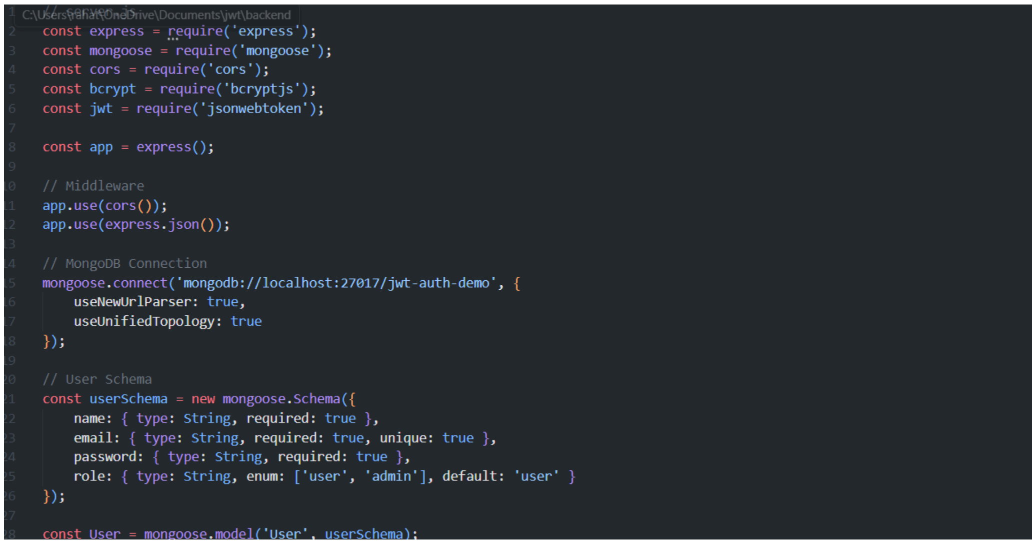


```
import { useState } from 'react';

export default function AuthApp() {
  const [view, setView] = useState('login');
  const [token, setToken] = useState(null);
  const [user, setUser] = useState(null);
  const [message, setMessage] = useState('');

  const [loginData, setLoginData] = useState({ email: '', password: '' });
  const [registerData, setRegisterData] = useState({
    name: '', email: '', password: '', role: 'user'
  });

  const handleRegister = () => {
    setMessage('');
    if (registerData.email && registerData.password && registerData.name) {
      setMessage('✅ Registration successful! Please login.');
      setView('login');
      setRegisterData({ name: '', email: '', password: '', role: 'user' });
    }
  };

  const handleLogin = () => {
    setMessage('');
    if (loginData.email === 'admin@test.com' && loginData.password === 'admin123') {
      const mockToken = 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.demo';
      const mockUser = { name: 'Admin User', email: 'admin@test.com', role: 'admin' };
      setToken(mockToken);
      setUser(mockUser);
      setView('dashboard');
      setMessage('');
    } else if (loginData.email === 'user@test.com' && loginData.password === 'user123') {
```

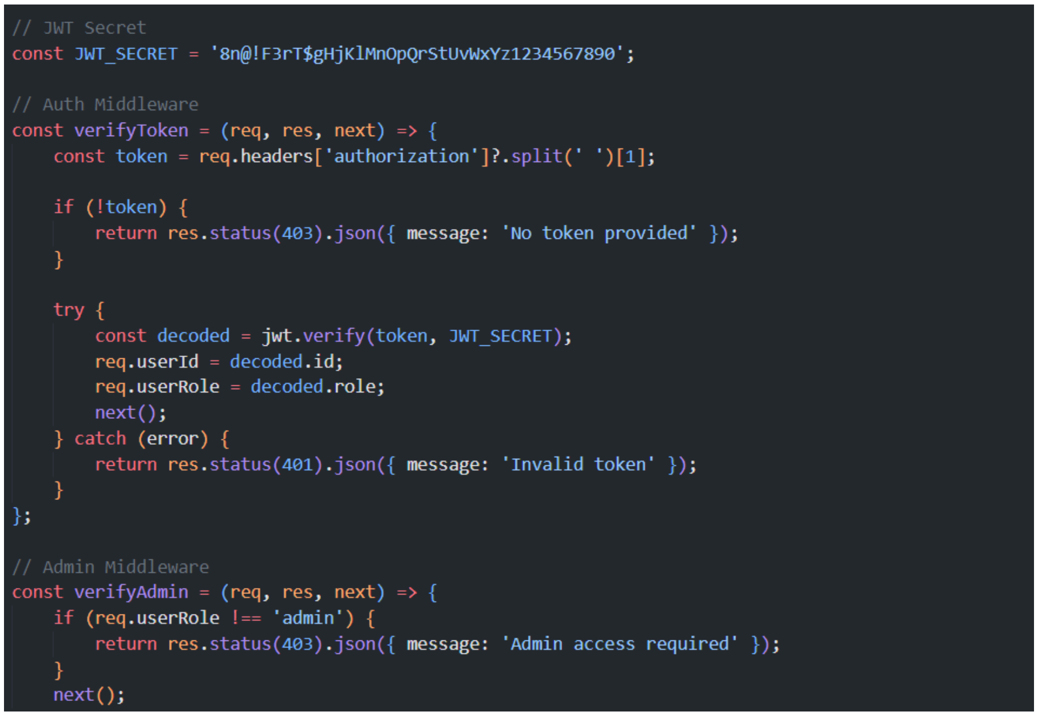


```
const handleLogout = () => {
  setToken(null);
  setUser(null);
  setView('login');
  setLoginData({ email: '', password: '' });
  setMessage('');
};

if (view === 'login') {
  return (
    <div className="min-h-screen bg-gradient-to-br from-blue-50 to-indigo-100 flex items-center justify-center p-4">
      <div className="bg-white rounded-2xl shadow-xl p-8 w-full max-w-md">
        <div className="text-center mb-8">
          <div className="bg-indigo-600 w-16 h-16 rounded-full flex items-center justify-center mx-auto 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
            </svg>
          </div>
          <h1 className="text-3xl font-bold text-gray-800">Login</h1>
          <p className="text-gray-600 mt-2">JWT Authentication Demo</p>
        </div>

        {message && (
          <div className="mb-4 p-3 bg-blue-50 border border-blue-200 rounded-lg text-sm text-blue-800">
            {message}
          </div>
        )}
```

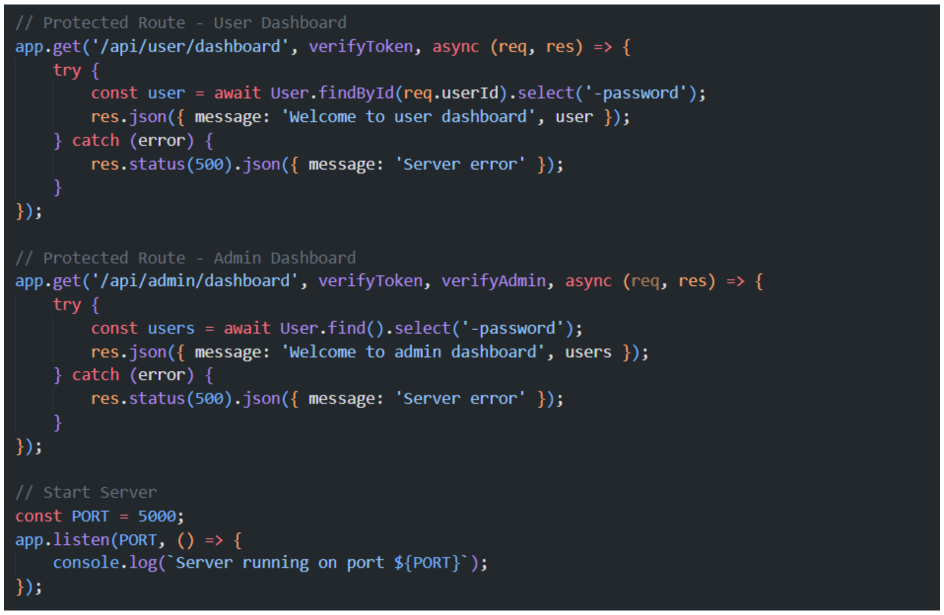
server.js



```
// server.js
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');

const app = express();

// Middleware
app.use(cors());
app.use(express.json());

// MongoDB Connection
mongoose.connect('mongodb://localhost:27017/jwt-auth-demo', {
    useNewUrlParser: true,
    useUnifiedTopology: true
});

// User Schema
const userSchema = new mongoose.Schema({
    name: { type: String, required: true },
    email: { type: String, required: true, unique: true },
    password: { type: String, required: true },
    role: { type: String, enum: ['user', 'admin'], default: 'user' }
});

const User = mongoose.model('User', userSchema);
```

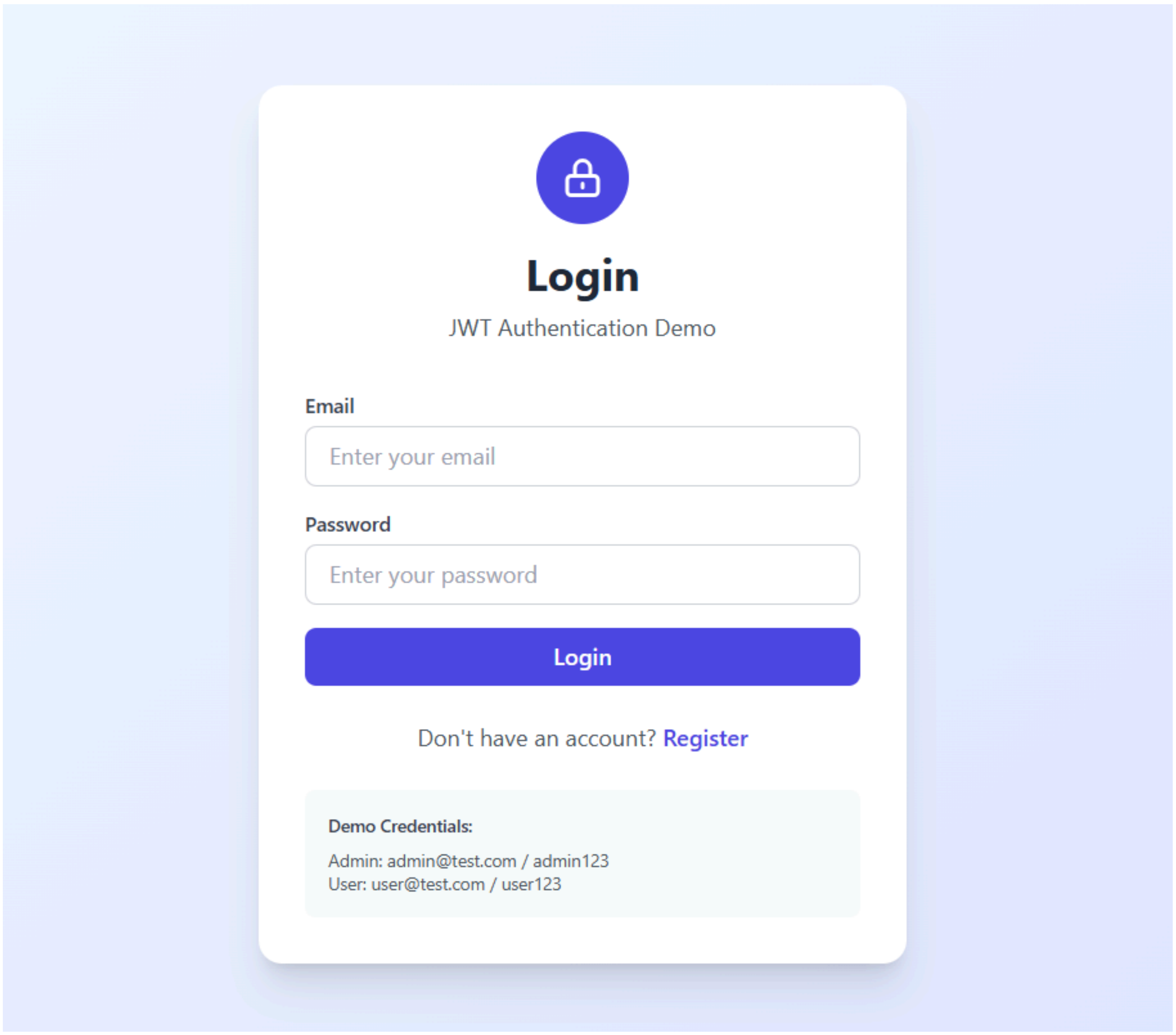


```
// JWT Secret
const JWT_SECRET = '8n@!F3rT$gHjKlMnOpQrStUvWxYz1234567890';

// Auth Middleware
const verifyToken = (req, res, next) => {
    const token = req.headers['authorization']?.split(' ')[1];

    if (!token) {
        return res.status(403).json({ message: 'No token provided' });
    }

    try {
        const decoded = jwt.verify(token, JWT_SECRET);
        req.userId = decoded.id;
        req.userRole = decoded.role;
        next();
    } catch (error) {
        return res.status(401).json({ message: 'Invalid token' });
    }
};

// Admin Middleware
const verifyAdmin = (req, res, next) => {
    if (req.userRole !== 'admin') {
        return res.status(403).json({ message: 'Admin access required' });
    }
    next();
```



```
// Protected Route - User Dashboard
app.get('/api/user/dashboard', verifyToken, async (req, res) => {
    try {
        const user = await User.findById(req.userId).select('-password');
        res.json({ message: 'Welcome to user dashboard', user });
    } catch (error) {
        res.status(500).json({ message: 'Server error' });
    }
});

// Protected Route - Admin Dashboard
app.get('/api/admin/dashboard', verifyToken, verifyAdmin, async (req, res) => {
    try {
        const users = await User.find().select('-password');
        res.json({ message: 'Welcome to admin dashboard', users });
    } catch (error) {
        res.status(500).json({ message: 'Server error' });
    }
});

// Start Server
const PORT = 5000;
app.listen(PORT, () => {
    console.log(`Server running on port ${PORT}`);
});
```

Output:



Login

JWT Authentication Demo

Email

Enter your email

Password

Enter your password

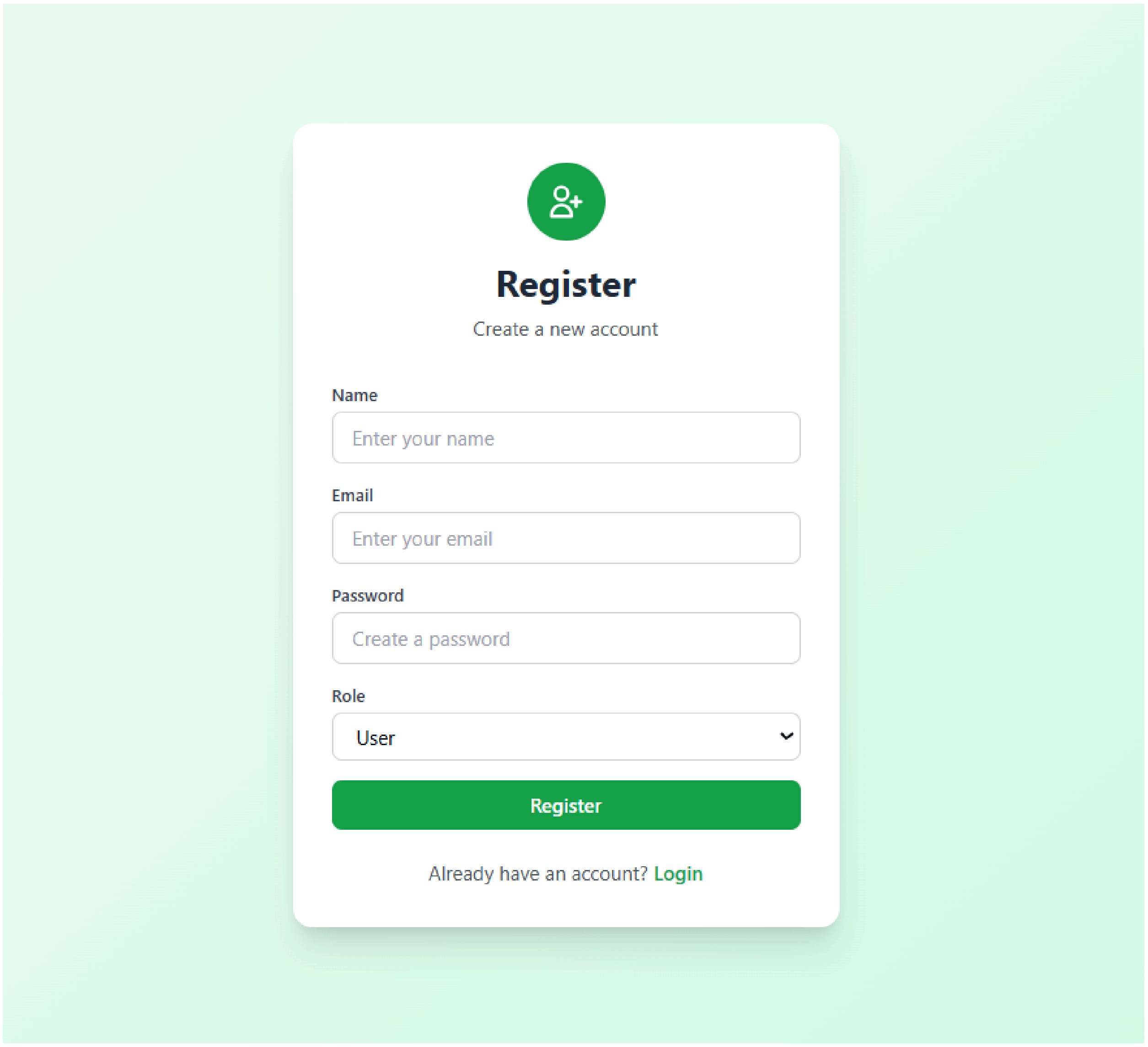
Login

Don't have an account? Register

Demo Credentials:

Admin: admin@test.com / admin123
User: user@test.com / user123

Login Board



Register

Create a new account

Name

Enter your name

Email

Enter your email

Password

Create a password

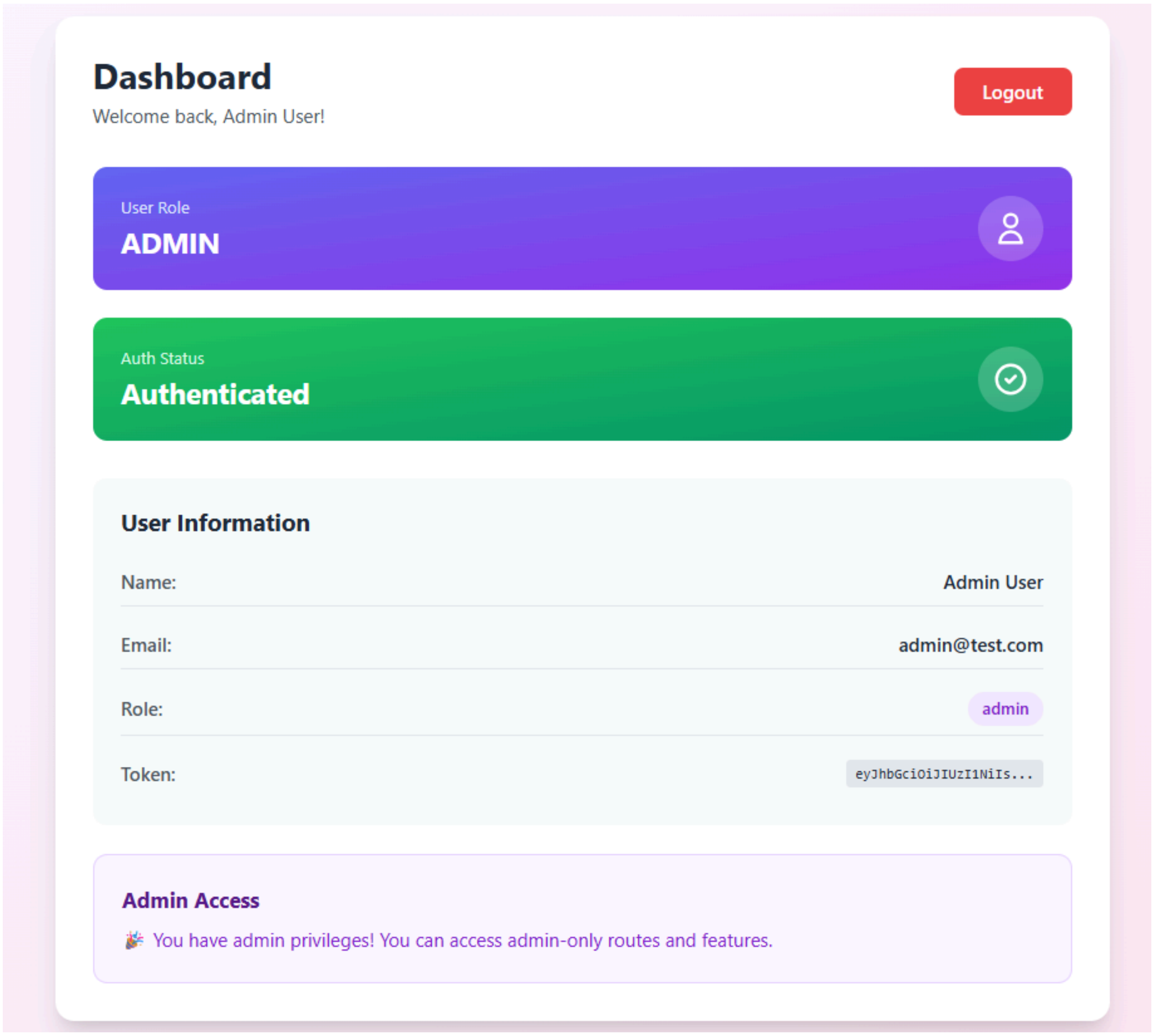
Role

User

Register

Already have an account? Login

Register Board:



Dashboard

Welcome back, Admin User!

User Role

ADMIN

Auth Status

Authenticated

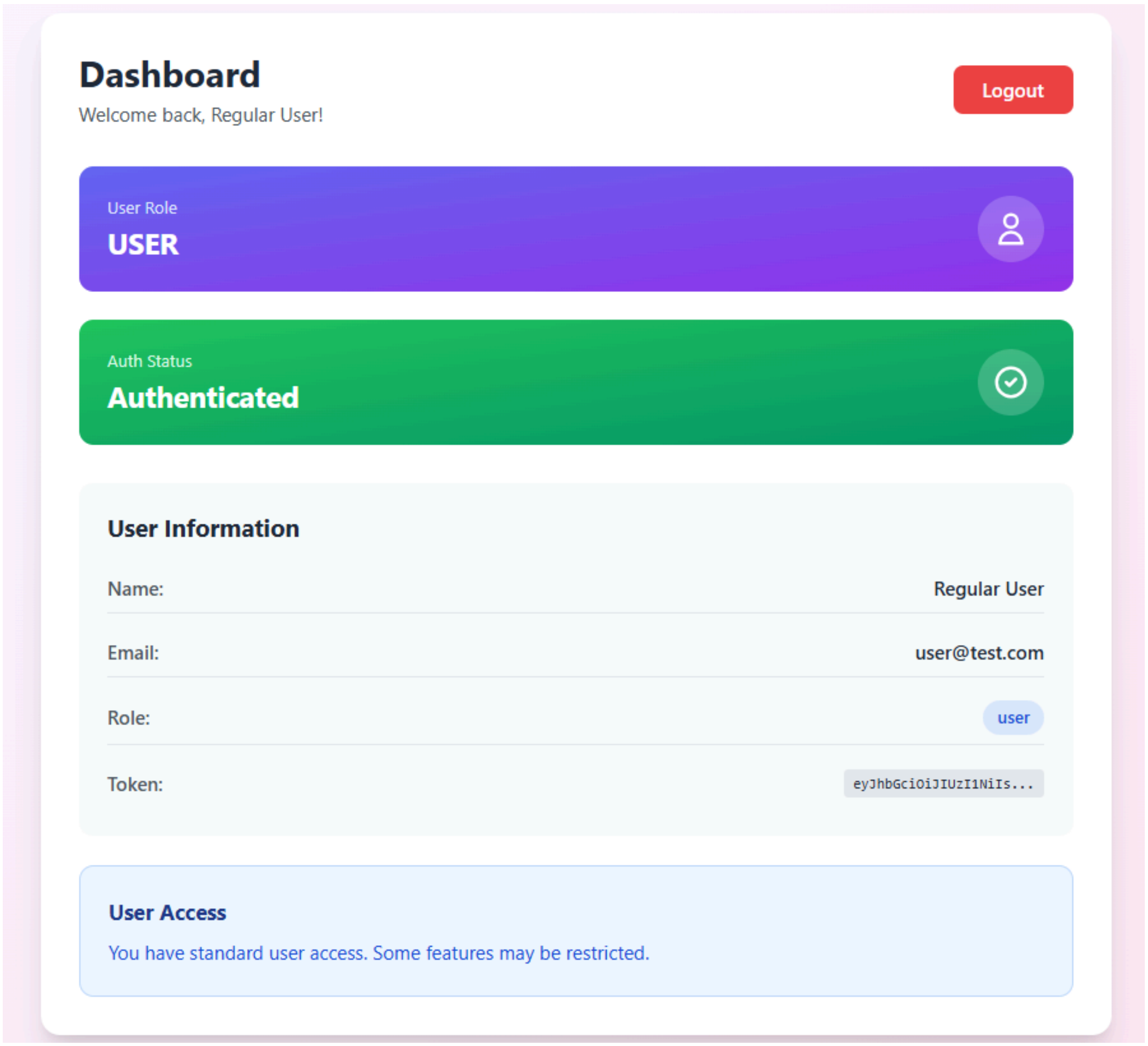
User Information

Name:	Admin User
Email:	admin@test.com
Role:	admin
Token:	eyJhbGciOiJIUzI1NiIs...

Admin Access

🎉 You have admin privileges! You can access admin-only routes and features.

Admin Panel



Dashboard

Welcome back, Regular User!

User Role

USER

Auth Status

Authenticated

User Information

Name:	Regular User
Email:	user@test.com
Role:	user
Token:	eyJhbGciOiJIUzI1NiIs...

User Access

You have standard user access. Some features may be restricted.

User Dashboard