Experiment-6.3

Build Role-Based Access Control with Admin, User, and Moderator Roles

Code-

RBAC Server (server.js)

This file contains the complete code for the Node.js server with Role-Based Access Control.

```
const express = require('express');
const jwt = require('jsonwebtoken');

const app = express();
const port = 3000;

app.use(express.json());

// --- Configuration ---
const JWT_SECRET = 'your-super-secret-key-that-is-very-long-and-secure';

// Expanded user "database" with roles
const users = [

     { id: 2, username: 'modUser', password: 'mod123', role: 'Moderator' },
    { id: 3, username: 'normalUser', password: 'user123', role: 'User' }
];

// --- 1. Login Route (Issues Token with Role) ---
// app.post('/api/login', (req, res) => {
    const { username, password } = req.body;
     const user = users.find(u => u.username === username && u.password ===


    if (!user) {
        return res.status(401).json({ message: 'Invalid username or password'  })

    }

    // Create the payload *with the user's role*
    const payload = {
        id: user.id,
        username: user.username,
        role: user.role // <-- This is the key change


    // Sign the token
    const token = jwt.sign(payload, JWT_SECRET, { expiresIn: '1h' });

    res.json({ token: token });
});

// --- 2. Authentication & Authorization Middleware ---

```

```javascript
/**
 * Middleware: Verify JWT (Authentication)
 * This is the same as before. It just checks if the token is valid.
 * It attaches the *entire* payload (id, username, role) to req.user.
 */
function verifyToken(req, res, next) {
    const authHeader = req.headers['authorization'];
    const token = authHeader && authHeader.split(' ')[1];

    if (token == null) {
        return res.status(401).json({ message: 'Token missing' });
    }

    jwt.verify(token, JWT_SECRET, (err, userPayload) => {
        if (err) {
            return res.status(403).json({ message: 'Token is invalid or expired' });
        }
        // Attach the payload (which includes the role) to the request
        req.user = userPayload;
        next();
    });
}

/**
 * Middleware: Check Role (Authorization)
 * This is the new middleware. It's a function that *returns* a middleware.
 * This allows us to pass in the roles we want to allow.
 */
function checkRole(allowedRoles) {
    return (req, res, next) => {
        // We assume verifyToken has already run, so req.user exists
        if (!req.user || !req.user.role) {
            return res.status(403).json({ message: 'Error: User role not found in token.' });
        }

        const userRole = req.user.role;

        // Check if the user's role is in the list of allowed roles
        if (!allowedRoles.includes(userRole)) {
            //              return res.status(403).json({ message: 'Access denied: insufficient role' });
        }

        // User has the correct role, proceed to the route
        next();
    };
}


// --- 3. Protected Routes with Role Checks ---

// Route 1: Accessible by all authenticated users (any role)
app.get('/api/user-profile', verifyToken, (req, res) => {
    //     // The checkRole middleware isn't needed here since all roles are allowed.
    // An Admin can see their own profile.
    res.json({
        message: `Welcome to your profile, ${req.user.username}`,
        user: req.user
    });
});

// Route 2: Accessible by Moderators AND Admins
app.get('/api/moderator-panel',
    verifyToken,                          // First, check if they are logged in
    checkRole(['Moderator', 'Admin']),    // Then, check if they have the 'Moderator' or 'Admin' role
    (req, res) => {
        res.json({
            message: 'Welcome to the Moderator Panel',
            user: req.user
        });
    });

// Route 3: Accessible ONLY by Admins
app.get('/api/admin-dashboard',
    verifyToken,                     // First, check if they are logged in
    checkRole(['Admin']),            // Then, check if they have the 'Admin' role
    (req, res) => {
        //         res.json({
            message: 'Welcome to the Admin Dashboard',
            user: req.user
        });
    });

// --- Start Server ---
app.listen(port, () => {
    console.log(`Server running on http://localhost:${port}`);
    console.log('---');
    console.log('Test users:');
    console.log('1. { username: "adminUser", password: "admin123", role: "Admin" }');
    console.log('2. { username: "modUser", password: "mod123", role: "Moderator" }');
    console.log('3. { username: "normalUser", password: "user123", role: "User" }');
    console.log('---');
});
```

Listing 1: server.js - RBAC Server