Experiment-6.2

Implement Protected Routes with JWT Verification

Code-

1. Node.js Server (server.js)

The main server file. It includes the login logic, middleware, and protected routes.

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

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

// Use Express's built-in middleware to parse JSON bodies
app.use(express.json());

// --- Configuration ---

// NEVER hardcode secrets in a real app. Use environment variables.
const JWT_SECRET = 'your-super-secret-key-that-is-very-long-and-secure';

// Hardcoded user database for this example
const users = [
    {
        id: 1,
        username: 'testuser',
        password: 'password123'
    }
];

// --- Routes ---

/**
 * Public route
 * Anyone can access this.
 */
app.get('/api/public', (req, res) => {
    res.json({ message: 'This route is public and open to everyone.' });
});

/**
 * Login route
 * Issues a JWT if credentials are valid.
 */
app.post('/api/login', (req, res) => {
    // Get username and password from request body
    const { username, password } = req.body;

    // Find the user in our "database"
    const user = users.find(u => u.username === username && u.password ===
    password);
```

```
44
45      if (!user) {
46          return res.status(401).json({ message: 'Invalid username or password'  })
47      }
48
49      // User is valid. Create a payload for the token.
50      // Don't include sensitive info like the password!
51      const payload = {
52          id: user.id,
53          username: user.username,
54      };
55
56      // Sign the token
57      const token = jwt.sign(
58          payload,
59          JWT_SECRET,
60          { expiresIn: '1h' } // Token expires in 1 hour
61
62
63      // Send the token to the client
64      res.json({ token: token });
65  });
66
67  /**
68   * Protected route
69   * Only accessible with a valid JWT.
70   */
71  app.get('/api/protected', verifyToken, (req, res) => {
72      // If the code reaches this point, it means the verifyToken middleware
73      // successfully authenticated the user.
74      // The user's data (from the token payload) is attached to req.user.
75      res.json({
76          message: 'You have accessed a protected route!',
77          user: req.user
78      });
79  });
80
81
82  // --- Middleware ---
83
84  /**
85
86   * This function intercepts requests to protected routes.
87   */
88  function verifyToken(req, res, next) {
89      // 1. Get the authorization header
90      const authHeader = req.headers['authorization'];
91
92      // 2. Check if it exists and has the 'Bearer ' prefix
93      // Format is "Bearer <token>"
94      const token = authHeader && authHeader.split(' ')[1];
95
96      // 3. If no token, send 401 Unauthorized
97      if (token == null) {
98          return res.status(401).json({ message: 'Token missing' });
99      }
100
```

```
    // 4. Verify the token
    jwt.verify(token, JWT_SECRET, (err, userPayload) => {
        if (err) {
            // If token is invalid (expired, wrong signature, etc.)
            return res.status(403).json({ message: 'Token is invalid or expired' });
        }

        // 5. Token is valid. Attach payload to request object
        req.user = userPayload;

        // 6. Call the next middleware or route handler
        next();
    });
}


// --- Start Server ---
app.listen(port, () => {
    console.log(`Server running on http://localhost:${port}`);
    console.log('---');
    console.log('Hardcoded user for testing:');
    console.log('Username: testuser');
    console.log('Password: password123');
    console.log('---');
    console.log('Routes:');
    console.log('POST /api/login (public)');
    console.log('GET /api/public (public)');
    console.log('GET /api/protected (protected)');
});
```

Listing 1: server.js - Main Application

2. Dependencies (package.json)

This file defines the project's dependencies, which are installed with 'npm install'.

```
{
  "name": "jwt-protected-routes",
  "version": "1.0.0",
  "description": "Example of JWT protected routes",
  "main": "server.js",
  "scripts": {
    "start": "node server.js"
  },
  "dependencies": {
    "express": "^4.19.2",
    "jsonwebtoken": "^9.0.2"
  }
}
```

Listing 2: package.json - Dependencies

3. Terminal Commands

These are the commands you run in your terminal to install and start the server.

```
# 1. Install dependencies
npm install

# 2. Start the server
npm start
```

Listing 3: Installation and Startup

4. Postman Login Body

This is the JSON data you send in the body of your POST request to '/api/login'.

```
{
    "username": "testuser",
    "password": "password123"
}
```

Listing 4: POST /api/login (raw, JSON)