Experiment-7.1

Middleware Implementation for Logging and Bearer Token Authentication

Code-

Middleware Server (server.js)

This file contains the complete code for the Node.js server with logging and Bearer token middleware.

```javascript
const express = require('express');
const app = express();
const port = 3000;

// --- 1. Logging Middleware (Global) ---
/**
 * This middleware function logs details for *every* request.
 * It's applied globally before any routes are defined.
 */
const requestLogger = (req, res, next) => {
    const timestamp = new Date().toISOString();
    console.log(`[${timestamp}] ${req.method} ${req.url}`);
    next(); // Pass control to the next middleware in the stack
};

// Apply the logging middleware globally
app.use(requestLogger);

// We still need this to parse JSON bodies, though not used in this example
app.use(express.json());

// --- 2. Bearer Token Authentication Middleware ---
/**
 * This middleware function checks for a specific Bearer token.
 * It will be applied *only* to our protected route.
 */
const verifyBearerToken = (req, res, next) => {
    const authHeader = req.headers['authorization'];

    // Check if the header exists
    if (!authHeader) {
        return res.status(401).json({ message: 'Authorization header missing' })
    ;
    }

    // Check if it's a Bearer token
    const parts = authHeader.split(' ');
    if (parts.length !== 2 || parts[0] !== 'Bearer') {
        return res.status(401).json({ message: 'Authorization header is
    malformed. Use Bearer <token>.' });
    }

    const token = parts[1];

    // Check if the token is the correct one
    if (token !== 'mysecrettoken') {
        return res.status(403).json({ message: 'Invalid token. Access forbidden.' });
    }

    // If all checks pass, proceed to the route
    next();
};


// --- 3. Routes ---

/**
 * Public Route
 * This route only has the global `requestLogger` middleware applied.
 */
app.get('/api/public', (req, res) => {
    res.json({ message: 'This is a public route. Anyone can see this!' });
});

/**
 * Protected Route
 * This route has *both* middlewares.
 * 1. `requestLogger` (from `app.use` above) runs first.
 * 2. `verifyBearerToken` (passed in here) runs second.
 * 3. The route handler `(req, res) => { ... }` runs last.
 */
app.get('/api/protected', verifyBearerToken, (req, res) => {
    res.json({ message: 'Success! You have accessed the protected route.' });
});


// --- Start Server ---
app.listen(port, () => {
    console.log(`Server running on http://localhost:${port}`);
    console.log('---');
    console.log('Test with the following routes:');
    console.log('1. GET http://localhost:3000/api/public (Public)');
    console.log('2. GET http://localhost:3000/api/protected (Protected)');
    console.log('   - Requires Header: "Authorization: Bearer mysecrettoken"');
    console.log('---');
    console.log('Watching for requests...');
});
```

Listing 1: server.js - Middleware Server