Experiment: Middleware Implementation for Logging and Bearer Token Authentication

AIM:

To implement middleware functions for logging HTTP requests and performing Bearer Token authentication in a Node.js Express application.

THEORY:

Middleware in Express.js is a function that has access to the request (req), response (res), and the next middleware function in the request-response cycle. They are primarily used for executing code before the request reaches the route handler, modifying the req and res objects, logging requests, handling authentication, and error handling.

1. Logging Middleware: This middleware logs the details of each incoming request such as method, URL, and timestamp. It helps in tracking API usage and debugging.
2. Bearer Token Authentication Middleware: This middleware checks the presence and validity of a Bearer Token in the Authorization header. A Bearer Token is usually a JWT (JSON Web Token) that allows secure access to protected resources by validating the token's authenticity.

PROCEDURE:

1. Initialize a new Node.js project using npm init.
2. Install Express using npm install express.
3. Create a file named app.js.
4. Implement logging and authentication middleware.
5. Define protected and public routes.
6. Test the API using Postman or any REST client.

CODE:

// app.js const express = require('express'); const app = express(); // Logging Middleware function logger(req, res, next) { console.log(`[${new Date().toISOString()}] ${req.method} ${req.url}`); next(); } // Bearer Token Authentication Middleware function authenticateToken(req, res, next) { const authHeader = req.headers['authorization']; const token = authHeader && authHeader.split(' ')[1]; if (!token) return res.status(401).json({ message: 'Access token missing' }); if (token === 'mysecrettoken') { next(); } else { res.status(403).json({ message: 'Invalid token' }); } } // Use Middlewares app.use(logger); // Public Route app.get('/', (req, res) => { res.send('Public Route - No Authentication Required'); }); // Protected Route app.get('/secure', authenticateToken, (req, res) => { res.send('Protected Route - Valid Token Provided'); }); // Start Server app.listen(3000, () => console.log('Server running on http://localhost:3000'));

EXPECTED OUTCOME:

- All incoming HTTP requests are logged with method, URL, and timestamp.
- Access to protected routes is allowed only when a valid Bearer Token is provided.
- Unauthorized access returns appropriate HTTP status codes (401 or 403).

LEARNING OUTCOMES:

- Understood the concept and importance of middleware in Express.js.
- Implemented custom middleware for logging and authentication.

- Gained practical knowledge of securing routes using Bearer Token validation.
- Learned how middleware enhances modularity and security in web applications.