■ Express Middleware Practice

1. Logger Middleware

Task: Create a middleware that logs the HTTP method, URL, and request time for every request.
Hint: Use req.method, req.url, and new Date().toLocaleTimeString(). Don’t forget next().
Expected Output: Visiting /about → console logs: GET /about at 12:30:45

2. Authentication Check

Task: Protect the /admin route: If query string has ?password=1234 → allow. Otherwise → 'Access Denied!'.
Hint: Use req.query.password.
Expected Output: /admin?password=1234 → Welcome Admin! /admin → Access Denied!

3. Add Data to Request

Task: Middleware should add req.requestTime = new Date(). Respond with this value in /time.
Hint: Attach data directly to req.
Expected Output: Visiting /time → Request time: Fri Sep 27 2025 12:45:32 GMT+0530

4. Multiple Middlewares

Task: Create two middlewares and a route: First logs 'First middleware executed', second logs 'Second middleware executed', final handler sends 'Done'.
Hint: Use app.use() or pass multiple middlewares to a route.
Expected Output: Console: First middleware executed, Second middleware executed Browser: Done

5. Time Restriction

Task: Write middleware that blocks requests between 12 AM – 6 AM with message 'Site closed. Try later.'.
Hint: Use new Date().getHours().
Expected Output: 2 AM → Site closed. Try later. 10 AM → normal route response.

6. Body Parser Simulation

Task: Make your own JSON body parser middleware (don’t use express.json()). Parse incoming request data and attach it to req.body.
Hint: Listen to req.on('data') and req.on('end').
Expected Output: POST /user with body {"name":"Ali"} → User name is Ali

7. Error Handling

Task: Create error-handling middleware: If any route throws an error → catch it and respond 'Something went wrong'.
Hint: Error-handling middleware must have 4 arguments: (err, req, res, next).

Expected Output: Throw an error in a route → Something went wrong

8. Role Check Middleware

Task: Create reusable middleware checkRole(role) that checks if query param matches the required role.
Hint: Call middleware like app.get('/admin', checkRole('admin'), ...).
Expected Output: /admin?role=admin → Welcome Admin! /admin?role=user → Not Authorized

9. Global vs Route Middleware

Task: Show difference between: Global middleware (runs on all requests). Route-specific middleware (runs only on /special).
Hint: Use app.use() for global and pass middleware only to /special for route-level.
Expected Output: /home → shows global middleware effect /special → shows both global + route middleware

10. Middleware Chain

Task: Build a chain of 3 middlewares: Logs request, adds req.user = { name: 'Test User' }, validates if req.user exists. Final response should say 'Welcome Test User'.
Hint: Middleware passes data via req and continues with next().
Expected Output: Visiting /profile → Welcome Test User