Practical – 12

GitHub Link: https://github.com/Devsharma511/NodeJs.git

Goal: Build a robust middleware stack with:

- X-Request-Id correlation
- High-precision timing
- Body size limits and JSON parse safety
- CORS (locked down to a whitelist)
- Per-route schema validation
- Centralized error handler returning RFC-7807 “problem + json”
- A simple endpoint to prove ordering works

Constraints:

- No unhandled rejections; async errors must hit the error handler.
- All responses include X-Request-Id and X-Response-Time-ms

Project Structure:

app.js:

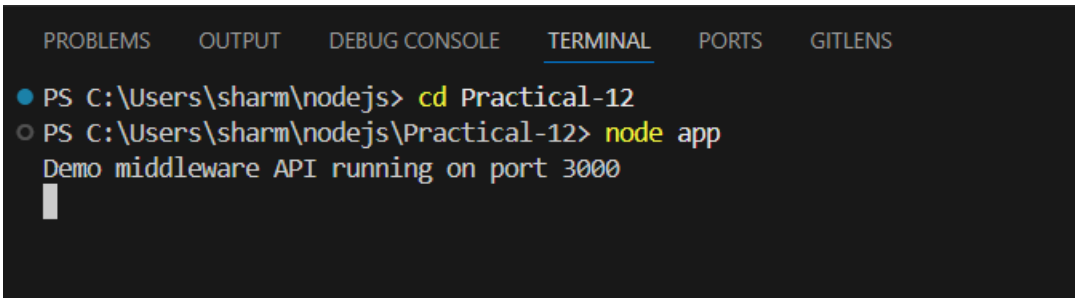
app.js U

Practical-12 > app.js > app.use() callback > res.on('finish') callback

```
import express from 'express';
import cors from 'cors';   5k (gzipped: 2.1k)

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

const allowedOrigins = ['http://localhost:5173', 'https://yourfrontend.example'];

app.use((req, res, next) => {
  req.id = req.headers['x-request-id'] || Math.random().toString(36).slice(2);
  res.setHeader('X-Request-Id', req.id);
  next();
});

app.use((req, res, next) => {
  const start = process.hrtime.bigint();

  res.on('finish', () => {
    const end = process.hrtime.bigint();
    const ms = Number(end - start) / 1e6;
    if (!res.headersSent) {
      res.setHeader('X-Response-Time-ms', ms.toFixed(2));
    }
  });
  next();
});

app.use(express.json({ limit: '10kb' }));
app.use((err, req, res, next) => {
  if (err instanceof SyntaxError || err.type === 'entity.too.large') {
    return next({ status: 400, title: 'Bad Request', detail: 'Invalid or too large JSON body' });
  }
  next(err);
});

app.use(cors({
  origin: (origin, callback) => {
    if (!origin || allowedOrigins.includes(origin)) callback(null, true);
    else callback(new Error('Not allowed by CORS'));
  }
}));

app.get('/', (req, res) => {
  return res.send('Welcome to the middleware demo');
});

const orderSchema = {
  required: ['item', 'quantity'],
  types: { item: 'string', quantity: 'number' }
};
```

```
function validateOrderSchema(req, res, next) {
  const { item, quantity } = req.body;
  if (typeof item !== 'string' || typeof quantity !== 'number') {
    return next({
      status: 422,
      title: 'Validation Error',
      detail: 'Order schema not matched: item(string), quantity(number) required'
    });
  }
  next();
}

app.post('/order', validateOrderSchema, (req, res) => {
  return res.json({ success: true, order: req.body });
});

app.use((err, req, res, next) => {
  if (res.headersSent) {
    return next(err);
  }
  res.status(err.status || 500).type('application/problem+json').json({
    type: 'about:blank',
    title: err.title || 'Internal Server Error',
    status: err.status || 500,
    detail: err.detail || err.message || 'Unknown error',
    instance: req.originalUrl,
    'request-id': req.id
  });
});

process.on('unhandledRejection', (reason) => {
  console.error('Unhandled Rejection', reason);
});

app.listen(PORT, () => console.log(`Demo middleware API running on port ${PORT}`));
```

Output:



Welcome to the middleware demo