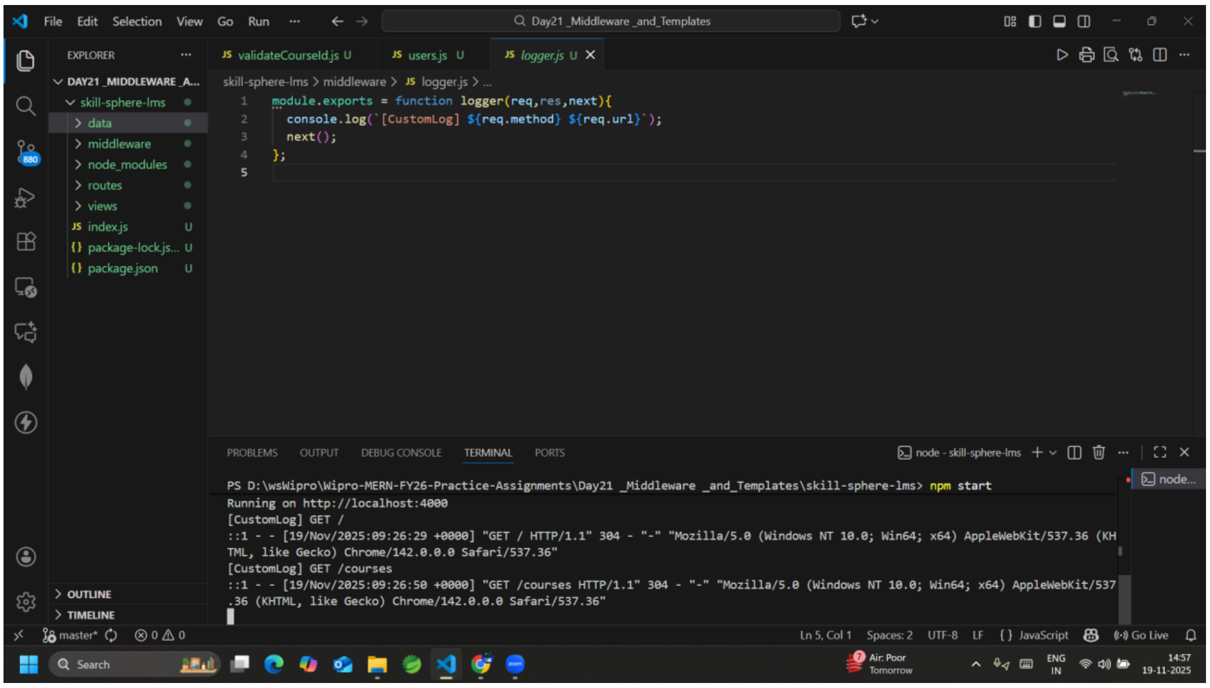
SkillSphere LMS — Day 21 Documentation Report

Middleware & Templates

User Story 1 – Global Logging Middleware

To monitor every incoming request, we implemented two logging middlewares: a lightweight custom logger and the Morgan production logger. The custom logger provides simple readable logs, while Morgan generates detailed HTTP logs suitable for analytics and debugging. This modular approach keeps logging clean and easily maintainable.

Output:



```
module.exports = function logger(req,res,next){
  console.log(`[CustomLog] ${req.method} ${req.url}`);
  next();
};

```

```
PS D:\wsWipro\Wipro-MERN-FY26-Practice-Assignments\Day21 _Middleware _and_Templates\skill-sphere-lms> npm start
Running on http://localhost:4000
[CustomLog] GET /
::1 - - [19/Nov/2025:09:26:29 +0000] "GET / HTTP/1.1" 304 - "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KH
TML, like Gecko) Chrome/142.0.0.0 Safari/537.36"
[CustomLog] GET /courses
::1 - - [19/Nov/2025:09:26:50 +0000] "GET /courses HTTP/1.1" 304 - "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537
.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36"
```

Code Snippet:

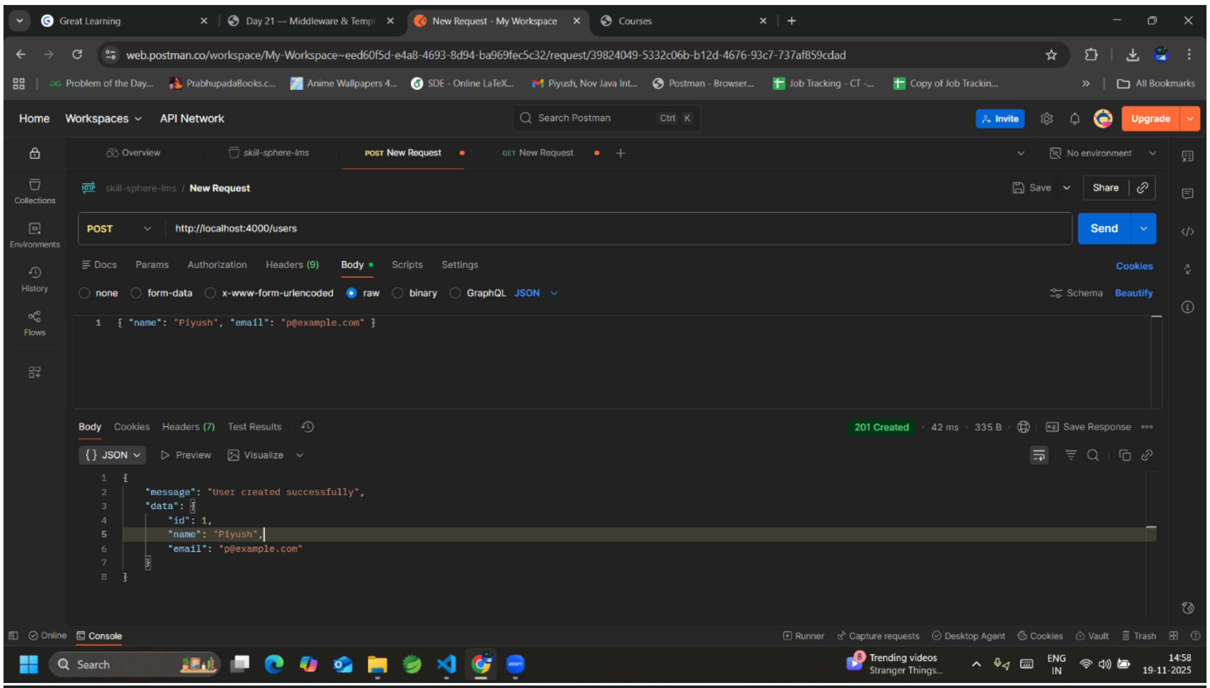
```
// Custom logger
module.exports = function logger(req,res,next){
  console.log(`[CustomLog] ${req.method} ${req.url}`);
  next();
};
app.use(logger);
```

```
// Morgan + custom logger
const morgan = require('morgan');
app.use(morgan('combined'));
```

User Story 2 – Body Parsing Middleware

To allow the server to accept JSON and form data, we enabled Express's built-in body parsing middleware. This ensures the backend can handle API requests and HTML form submissions without additional parsing logic. It also keeps routes clean and focused on business logic.

Output:

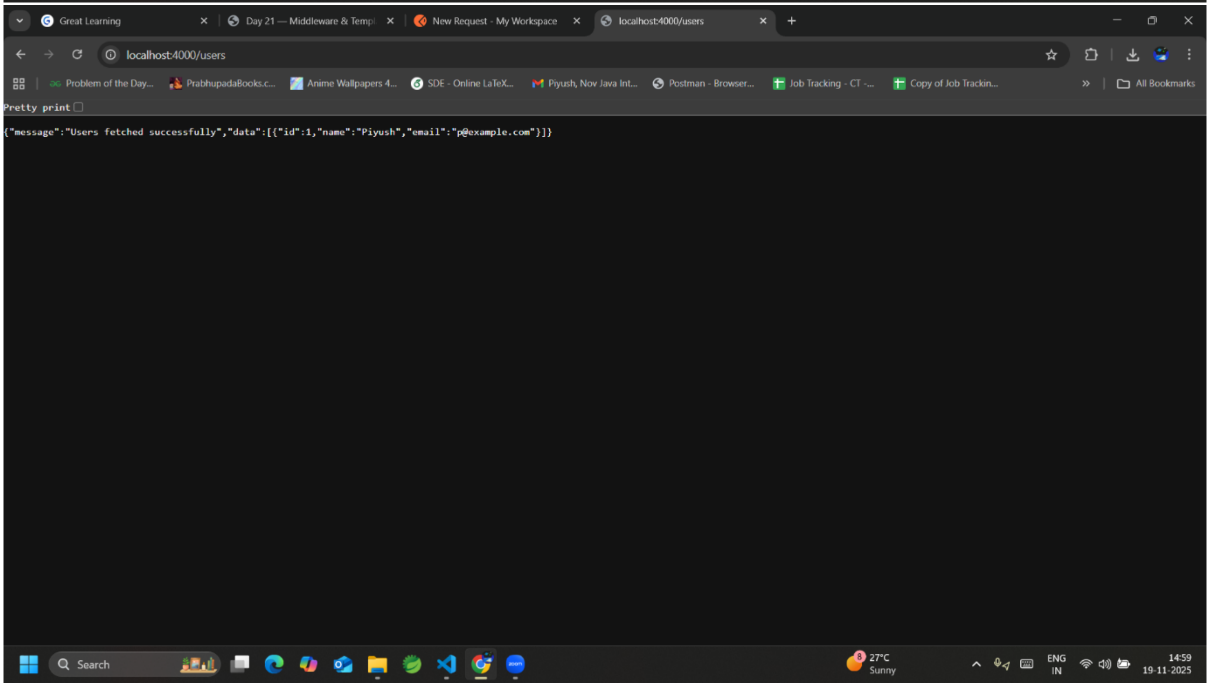


POST http://localhost:4000/users

```
{ "name": "Piyush", "email": "p@example.com" }
```

201 Created · 42 ms · 335 B

```
{
    "message": "User created successfully",
    "data": {
        "id": 1,
        "name": "Piyush",
        "email": "p@example.com"
    }
}
```



localhost:4000/users

```
{"message":"Users fetched successfully","data":[{"id":1,"name":"Piyush","email":"p@example.com"}]}
```

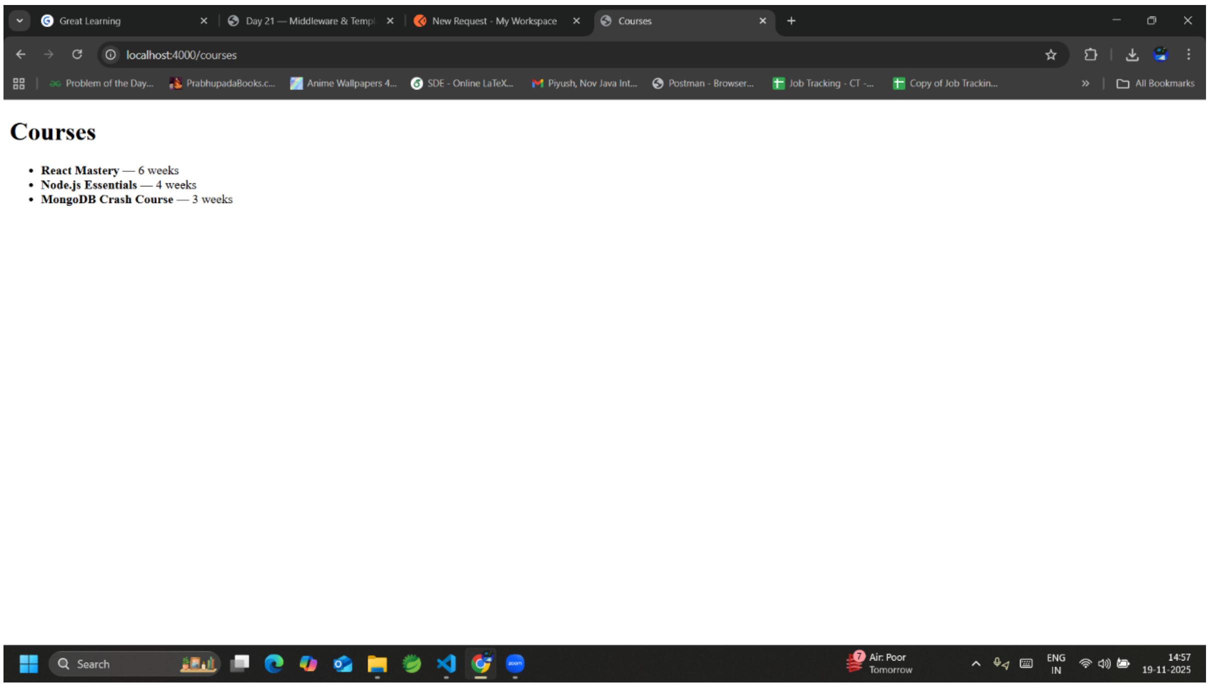
Code Snippet:

```
const express = require('express');
// Body parsers
app.use(express.json());
app.use(express.urlencoded({extended:true}));
```

User Story 3 – Render Courses Using EJS

To display course data visually, EJS was used as the templating engine. The route passes clean data to the template, ensuring the UI remains logic-free and easy to maintain. This separation of concerns keeps code modular and presentation-focused.

Output:



Courses

- **React Mastery** — 6 weeks
- **Node.js Essentials** — 4 weeks
- **MongoDB Crash Course** — 3 weeks

Code Snippet:

```
// View engine
const path = require('path');
app.set('view engine','ejs');
app.set('views', path.join(__dirname,'views'));
```

```
// views/courses.ejs
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8"/>
 <title>Courses</title>
</head>
<body>
 <h1>Courses</h1>
 <ul>
   <% courses.forEach(c=>{ %>
     <li><strong><%= c.name %></strong> — <%= c.duration %></li>
   <% }) %>
 </ul>
</body>
```

```
</html>
```