**Experiment Number: 8 Date: 24-03-2025**

# **Explore and Integrate Third-Party Node.js Extensions into an Express App**

## **PRE LAB EXERCISE**

### **Objective:**

* Understand the importance of **third-party extensions (middleware, libraries, and modules)** in Express.js applications.
* Learn how to integrate popular **Node.js extensions** to enhance functionality.
* Implement middleware for **logging, security, and API request handling**.

### **QUESTIONS:**

1. **What are third-party extensions in Node.js, and why are they useful?**

Third-party extensions in Node.js are external modules or packages that extend the functionality of a Node.js application. They are typically installed via **npm (Node Package Manager)** and help developers save time by providing pre-built solutions for common tasks such as routing, authentication, database management, and API handling.

1. **What is the purpose of Express middleware?**

Express middleware functions are used to handle request and response objects in an Express application. They can be used for **logging, authentication, error handling, request parsing, and more**. Middleware functions execute in the order they are defined and can modify the request, response, or pass control to the next function in the stack.

1. **How do you install and use third-party Node.js modules?**

* Install a module using npm: (sh)

npm install package-name

* Import and use it in your Node.js application: (js)

const packageName = require('package-name');

* Example (installing and using express): (sh)

npm install express

  //(js)

const express = require('express');

const app = express();

1. **Name some commonly used third-party packages in an Express.js application.**

* **express** – Web framework for Node.js
* **cors** – Handles Cross-Origin Resource Sharing
* **dotenv** – Loads environment variables from a .env file
* **morgan** – HTTP request logger
* **helmet** – Security middleware for setting HTTP headers
* **jsonwebtoken (jwt)** – Authentication using JSON Web Tokens
* **bcrypt** – Password hashing

1. **What is the difference between built-in middleware and third-party middleware in Express.js?**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Built-in Middleware** | **Third-party Middleware** |
| **Definition** | Middleware functions provided by Express.js itself. | Middleware developed by the community and installed via npm. |
| **Installation** | No installation required; comes with Express. | Requires installation using npm install package-name. |
| **Examples** | - express.json() (parses JSON request bodies)  - express.urlencoded() (parses URL-encoded data)  - express.static() (serves static files) | - cors (handles Cross-Origin Resource Sharing)  - morgan (HTTP request logger)  - helmet (enhances security) |
| **Usage** | Used directly in Express applications without extra setup. | Imported separately and added as middleware in the application. |

## **IN LAB EXERCISE**

### **Objective:**

* Install and use **third-party Node.js packages** in an Express application.
* Implement middleware for **logging (morgan), security (helmet), and request parsing (body-parser)**.
* Handle **CORS** (Cross-Origin Resource Sharing).

### **Resources Required:**

* **Node.js**, **Express.js**, **Morgan (Logger)**, **Helmet (Security)**, **Body-Parser (Request Parsing)**, **CORS Middleware**.

### **Step 1: Install Required Packages**

npm init -y

npm install express morgan helmet body-parser cors

A screen shot of a computer

AI-generated content may be incorrect.

### **Step 2: Set Up Express Server**

**Create server.js**

const express = require('express');

const morgan = require('morgan');

const helmet = require('helmet');

const bodyParser = require('body-parser');

const cors = require('cors');

const app = express();

// Middleware Setup

app.use(morgan('dev'));  // Logging HTTP requests

app.use(helmet());       // Security middleware

app.use(bodyParser.json());  // Parse JSON request bodies

app.use(cors());         // Enable CORS for cross-origin requests

// Sample Route

app.get('/', (req, res) => {

    res.send('Welcome to the Express Server with Third-Party Extensions!');

});

// API Route to test Body-Parser

app.post('/data', (req, res) => {

    res.json({ message: 'Data received successfully', data: req.body });

});

app.listen(3000, () => console.log('Server running on http://localhost:3000'));

### **Step 3: Explanation of Third-Party Modules Used**

1. **Morgan (HTTP Request Logger)**
   * Helps in **logging incoming requests** for debugging.
   * Logs request details like **method, URL, status, response time**.
2. **Helmet (Security Middleware)**
   * Protects against common security vulnerabilities like **Cross-Site Scripting (XSS)** and **Clickjacking**.
3. **Body-Parser (Request Parsing Middleware)**
   * Parses incoming **JSON data** from POST requests.
   * Without this, req.body would be undefined.
4. **CORS (Cross-Origin Resource Sharing)**
   * Allows API access from **different domains**.
   * Prevents **CORS policy issues** when making frontend requests.

### **Step 4: Run and Test the Server**

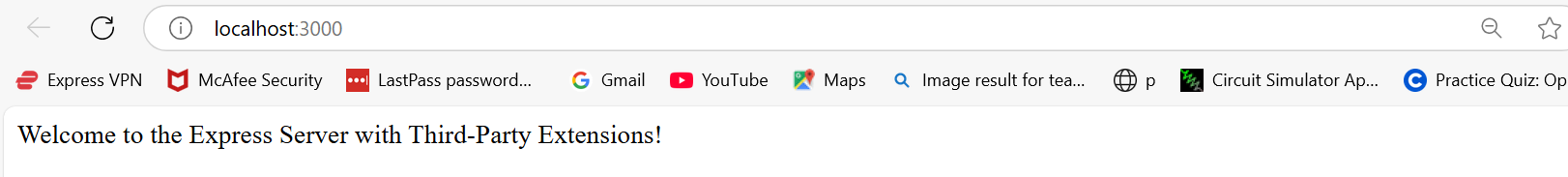
1. Start the server:

node server.js

1. Open a browser and visit:

http://localhost:3000

* + You should see: **"Welcome to the Express Server with Third-Party Extensions!"**

****

Test **Body-Parser Middleware** using Postman:

* + **POST request** to http://localhost:3000/data
  + **Body (JSON format):**

{

"name": "Kritti",

"age": 20

}

* + **Response:**

{

"message": "Data received successfully",

"data": {

"name": "Kritti",

"age": 20

}

}

A screenshot of a computer

AI-generated content may be incorrect.

## **POST LAB EXERCISE**

### **QUESTIONS:**

1. **What are the benefits of using third-party middleware in an Express app?**

Third-party middleware enhances an Express app by providing pre-built functionality, reducing development time and effort. Benefits include:

* **Security** – Middleware like helmet protects against vulnerabilities.
* **Logging & Debugging** – morgan helps track incoming requests.
* **Data Parsing** – body-parser simplifies handling JSON and form data.
* **Cross-Origin Requests** – cors enables safe API access from different origins.
* **Authentication** – passport and jsonwebtoken handle user authentication.

1. **How does Morgan help in debugging an application?**

Morgan is an HTTP request logger middleware that helps developers:

* Log incoming requests with details like method, URL, response status, and response time.
* Identify slow or failing requests.
* Save logs for debugging later.
* Use different log formats (combined, dev, common, etc.) to customize output.  
  **Example usage:**(js)

const morgan = require('morgan');

app.use(morgan('dev')); // Logs requests in a readable format

1. **Why is Helmet important for web security?**

Helmet secures Express apps by setting HTTP headers to prevent attacks like:

* **Cross-site scripting (XSS)**
* **Clickjacking**
* **Content sniffing**
* **Injection attacks**
* **Information leakage**

**Example usage:** (js)

const helmet = require('helmet');

app.use(helmet()); // Enables default security headers

1. **What is the role of Body-Parser, and why do we need it?**

Body-Parser is middleware used to parse incoming request bodies, making it easier to work with JSON and form data.

* Parses **JSON** payloads (express.json()).
* Parses **URL-encoded** data (express.urlencoded({ extended: true })).
* Required for processing POST and PUT requests with data.

**Example usage:** (js)

app.use(express.json()); // Parses JSON request bodies

app.use(express.urlencoded({ extended: true })); // Parses form data

1. **How does CORS help in web applications?**

CORS (Cross-Origin Resource Sharing) allows web applications to access resources from different origins.

* Prevents **CORS policy errors** when making API requests from different domains.
* Enables **secure data sharing** between front-end and back-end.
* Supports **custom headers and authentication mechanisms**.
* **Example usage:** (js)

const cors = require('cors');

app.use(cors()); // Allows all origins

app.use(cors({ origin: 'https://example.com' })); // Restricts to a specific domain

**ASSESSMENT PATTERN.**

|  |  |  |
| --- | --- | --- |
| **Description** | **Max Marks** | **Marks Awarded** |
| Pre Lab Exercise | **5** |  |
| In Lab Exercise | **10** |  |
| Post Lab Exercise | **5** |  |
| Viva | **10** |  |
| **Total** | **30** |  |
| **Faculty Signature** | |  |