1. Aim

To configure Express settings and create Express applications using request and response objects.

2. Requirements

Software Requirements:

- Node.js (Latest LTS version)
- npm (Node Package Manager)
- Express.js
- Code Editor (VS Code, Sublime Text, or any preferred IDE)

Hardware Requirements:

- Internet connection for installing dependencies

3. Theory

Introduction to Express.js

Express.js is a lightweight and flexible **Node.js framework** used to build **web applications and APIs**. It simplifies the process of handling **HTTP requests and responses**.

Key Features of Express.js

- **Middleware support** for handling requests, responses, and routing.
- **Routing mechanism** for defining application routes.
- **Template engines** for rendering dynamic content.
- **Error handling** for debugging applications.

Understanding Request and Response Objects

- The **request object (`req`)** contains information about the incoming HTTP request, such as parameters, headers, and body data.
- The **response object (`res`)** is used to send data back to the client.

Configuring Express Settings

- Express settings can be configured using the `app.set()` method.
- Example settings:
  - `app.set('view engine', 'ejs')` → Configures the template engine.
  - `app.set('port', 3000)` → Defines the port number.

---

4. Code

Step 1: Install Express

Before running the application, install Express using npm.

```
npm init -y
npm install express
```

Step 2: Create `server.js` and Configure Express

```
// Import Express module
const express = require('express');

// Create an Express application
const app = express();

// Set port number
const PORT = 3001;

// Middleware to parse JSON data
app.use(express.json());

// Define a GET route
app.get('/', (req, res) => {
    res.send('Welcome to Express Application!');
});

// Define a route to demonstrate request parameters
app.get('/user/:name', (req, res) => {
    const userName = req.params.name;
    res.send(`Hello, ${userName}!`);
});

// Define a POST route to handle user data
app.get('/data', (req, res) => {
    res.json({ message: "Use POST request to send data" });
});


// Start the Express server
app.listen(PORT, () => {
    console.log(`Server is running on http://localhost:${PORT}`);
});
```

5. Conclusion

In this practical, we configured Express settings and created a simple Express application. We explored the **request and response objects**, implemented **GET and POST routes**, and handled dynamic data. This knowledge is fundamental for developing **web applications and RESTful APIs** using Express.js.