4. Develop a basic Node.js server using the built-in http module to serve a JSON response. Extend it using Express.js to build a full REST API for product management. Use Postman for testing all endpoints

Node-http-server.js

```
const http = require('http');

const products = [
  { id: 1, name: 'Laptop', price: 1200 },
  { id: 2, name: 'Mouse', price: 25 },
  { id: 3, name: 'Keyboard', price: 75 }
];

// This is the function that will handle all incoming requests.
const requestListener = function (req, res) {
  // Set the response header to indicate that we're sending JSON data.
  res.setHeader('Content-Type', 'application/json');

  // Check the request URL to determine the response.
  if (req.url === '/api/products') {
    // If the URL is '/api/products', send back the list of products.
    res.writeHead(200); // 200 is the HTTP status code for OK.
    res.end(JSON.stringify(products));
  } else {
    // If the URL is anything else, send a "Not Found" response.
    res.writeHead(404); // 404 is the HTTP status code for Not Found.
    res.end(JSON.stringify({ message: 'Resource not found' }));
  }
};
// Create the server instance.
const server = http.createServer(requestListener);

// Define the port and host for the server to listen on.
const host = 'localhost';
const port = 3000;

// Start the server and log a message to the console.
server.listen(port, host, () => {
  console.log(`Server is running on http://${host}:${port}`);

});
```

Express-api.js

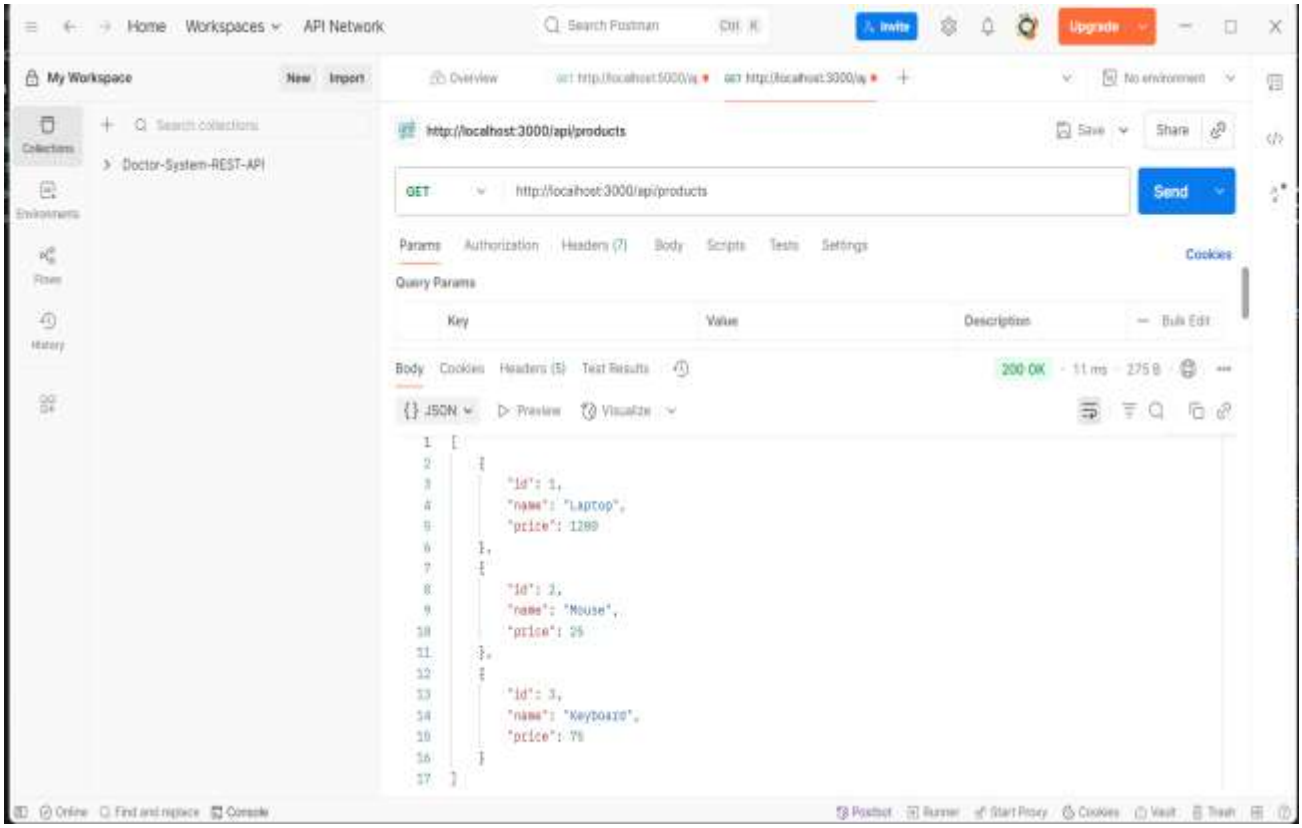
```
const express = require('express');
// Create an Express application instance.
```

```
const app = express();

// Enable the express app to parse JSON formatted request bodies.
// This is necessary to handle POST requests with JSON data.
app.use(express.json());

// Define a simple "database" of products in memory.
let products = [
  { id: 1, name: 'Laptop', price: 1200 },
  { id: 2, name: 'Mouse', price: 25 },
  { id: 3, name: 'Keyboard', price: 75 }
];

//--- GET ALL PRODUCTS ---
// This route handles GET requests to the /api/products endpoint.
app.get('/api/products', (req, res) => {
  // Send the entire products array as a JSON response.
  res.json(products);
});

//--- GET PRODUCT BY ID ---
// This route uses a route parameter (:id) to get a single product.
app.get('/api/products/:id', (req, res) => {
  // Find the product with a matching ID. The req.params.id is a string,
  // so we use parseInt to convert it to a number for comparison.
  const product = products.find(p => p.id === parseInt(req.params.id));

  // If the product is not found, send a 404 status with an error message.
  if (!product) {
    return res.status(404).json({ message: 'Product not found.' });
  }

  // If the product is found, send it as a JSON response.
  res.json(product);
});

//--- CREATE NEW PRODUCT ---
// This route handles POST requests to create a new product.
app.post('/api/products', (req, res) => {
  // Check if the request body contains a name and price.
  if (!req.body.name || !req.body.price) {
    return res.status(400).json({ message: 'Name and price are required.' });
  }

  // Create a new product object with a unique ID and the data from the
request body.
  const newProduct = {
    id: products.length > 0 ? products[products.length - 1].id + 1 : 1,
```

```
    name: req.body.name,
    price: req.body.price
  };

  // Add the new product to our in-memory array.
  products.push(newProduct);

  // Send back the new product object with a 201 Created status.
  res.status(201).json(newProduct);
});

// Define the port for the server to listen on.
const port = 5000;

// Start the server and log a message to the console.
app.listen(port, () => {
  console.log(`Express.js API is running on http://localhost:${port}`);
 });
```

OUTPUT:

Node-http-server.js



Express-api.js