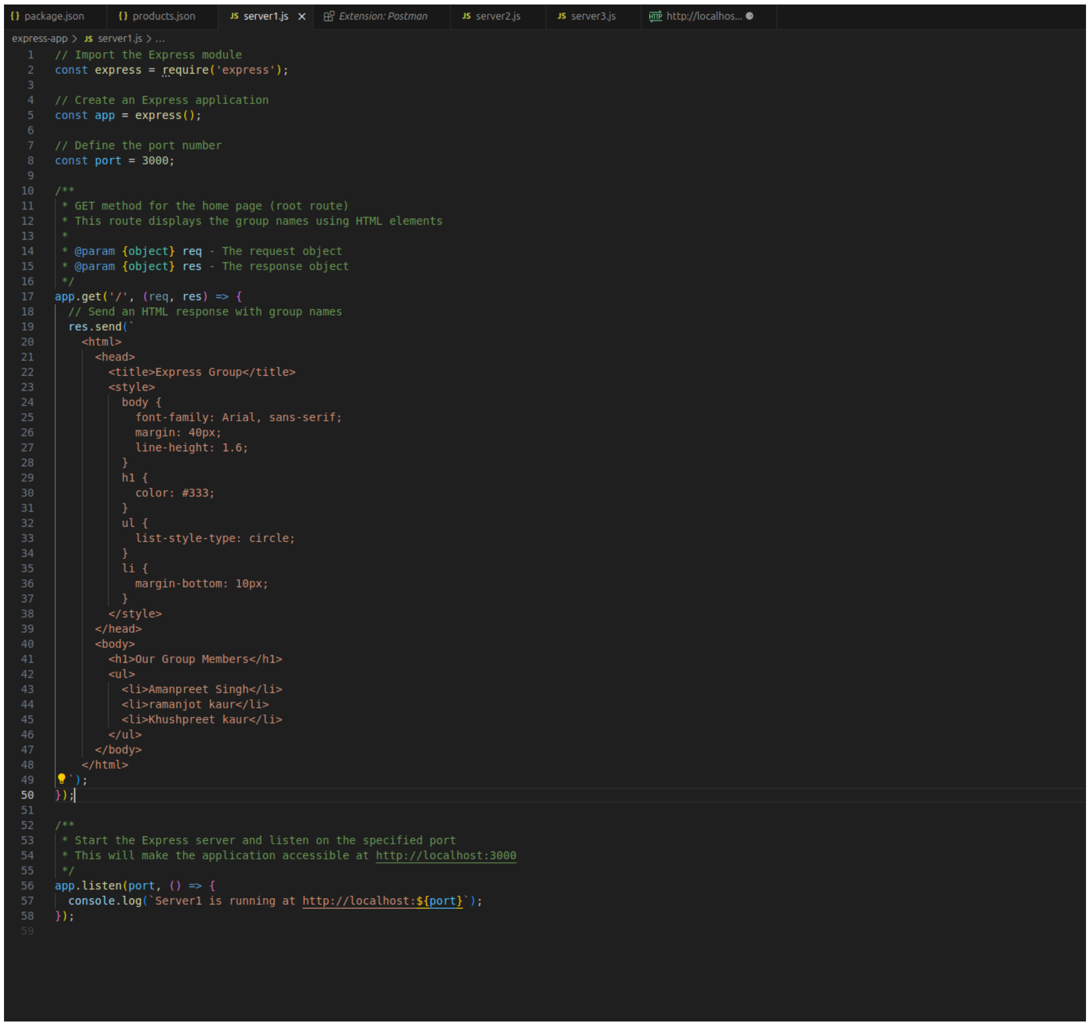
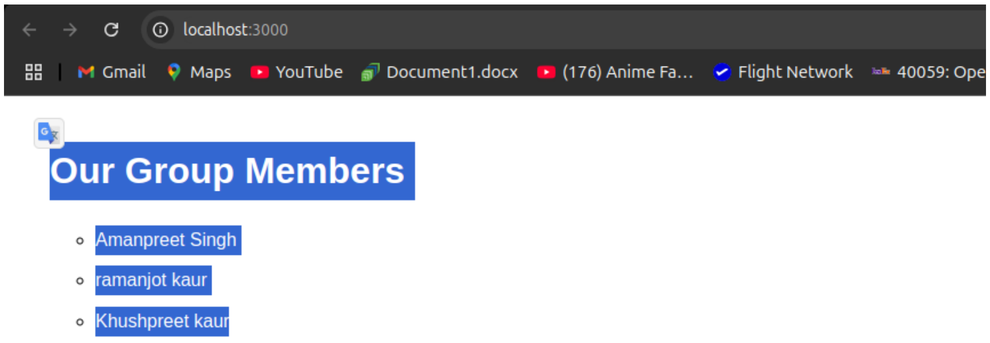
File 1:



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

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

// Define the port number
const port = 3000;

/**
 * GET method for the home page (root route)
 * This route displays the group names using HTML elements
 *
 * @param {object} req - The request object
 * @param {object} res - The response object
 */
app.get('/', (req, res) => {
  // Send an HTML response with group names
  res.send(`
    <html>
      <head>
        <title>Express Group</title>
        <style>
          body {
            font-family: Arial, sans-serif;
            margin: 40px;
            line-height: 1.6;
          }
          h1 {
            color: #333;
          }
          ul {
            list-style-type: circle;
          }
          li {
            margin-bottom: 10px;
          }
        </style>
      </head>
      <body>
        <h1>Our Group Members</h1>
        <ul>
          <li>Amanpreet Singh</li>
          <li>ramanjot kaur</li>
          <li>Khushpreet kaur</li>
        </ul>
      </body>
    </html>
  `);
});

/**
 * Start the Express server and listen on the specified port
 * This will make the application accessible at http://localhost:3000
 */
app.listen(port, () => {
  console.log(`Server1 is running at http://localhost:${port}`);
});
```

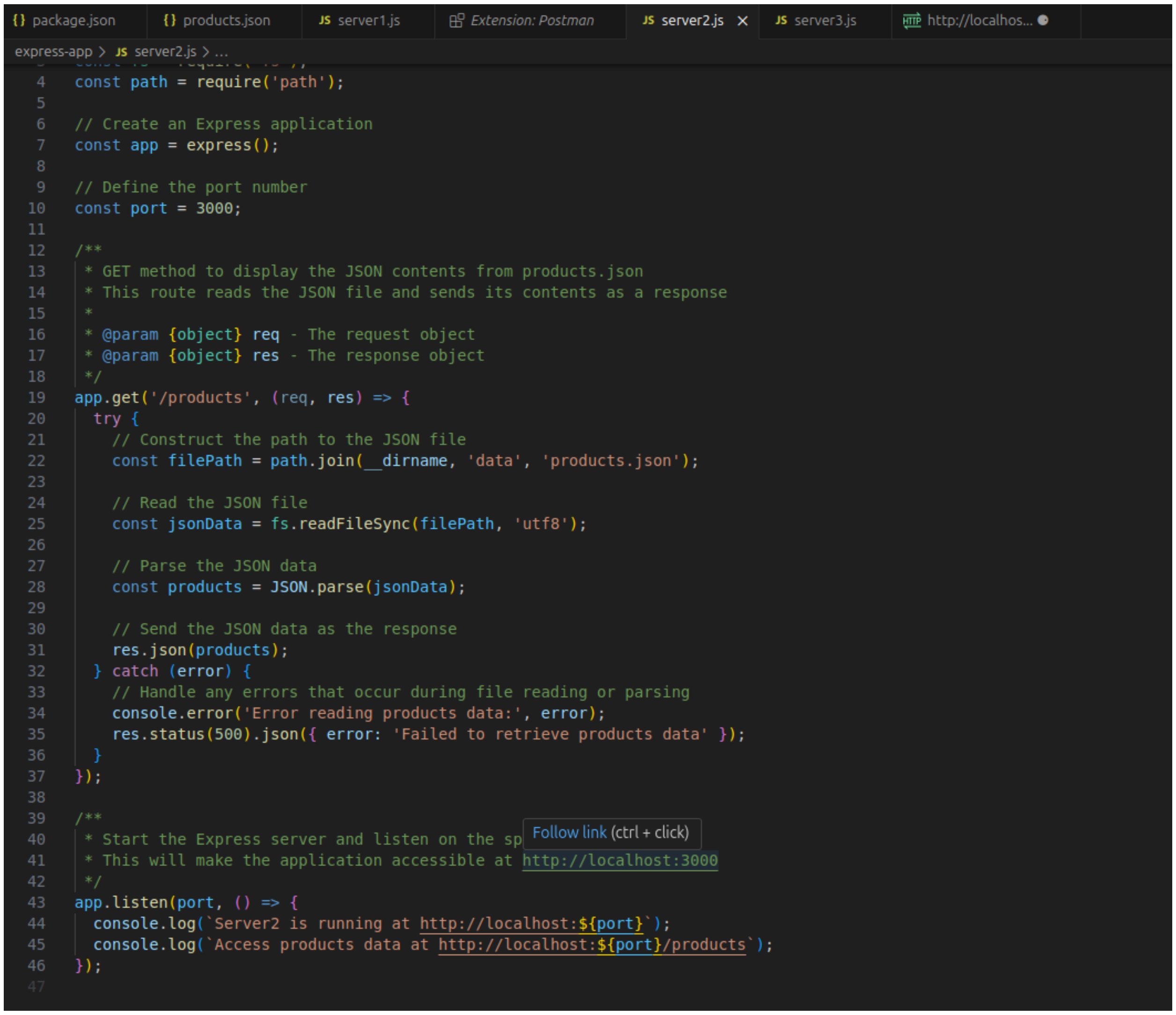
Response file 1 :



Our Group Members

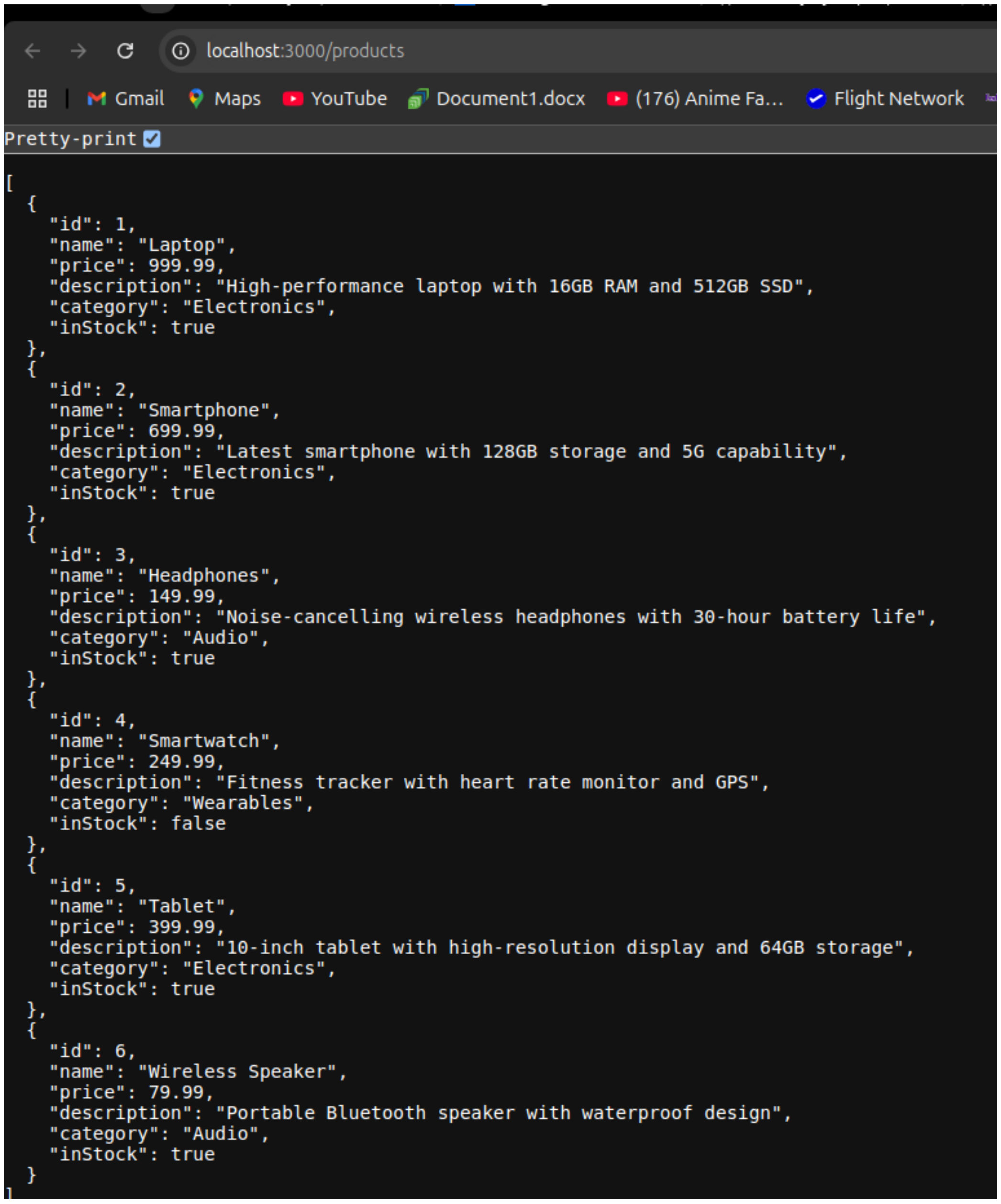
- Amanpreet Singh
- ramanjot kaur
- Khushpreet kaur

File 2:



```javascript
const path = require('path');

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

// Define the port number
const port = 3000;

/**
 * GET method to display the JSON contents from products.json
 * This route reads the JSON file and sends its contents as a response
 *
 * @param {object} req - The request object
 * @param {object} res - The response object
 */
app.get('/products', (req, res) => {
  try {
    // Construct the path to the JSON file
    const filePath = path.join(__dirname, 'data', 'products.json');

    // Read the JSON file
    const jsonData = fs.readFileSync(filePath, 'utf8');

    // Parse the JSON data
    const products = JSON.parse(jsonData);

    // Send the JSON data as the response
    res.json(products);
  } catch (error) {
    // Handle any errors that occur during file reading or parsing
    console.error('Error reading products data:', error);
    res.status(500).json({ error: 'Failed to retrieve products data' });
  }
});

/**
 * Start the Express server and listen on the sp Follow link (ctrl + click)
 * This will make the application accessible at http://localhost:3000
 */
app.listen(port, () => {
  console.log(`Server2 is running at http://localhost:${port}`);
  console.log(`Access products data at http://localhost:${port}/products`);
});
```

Response file 2:

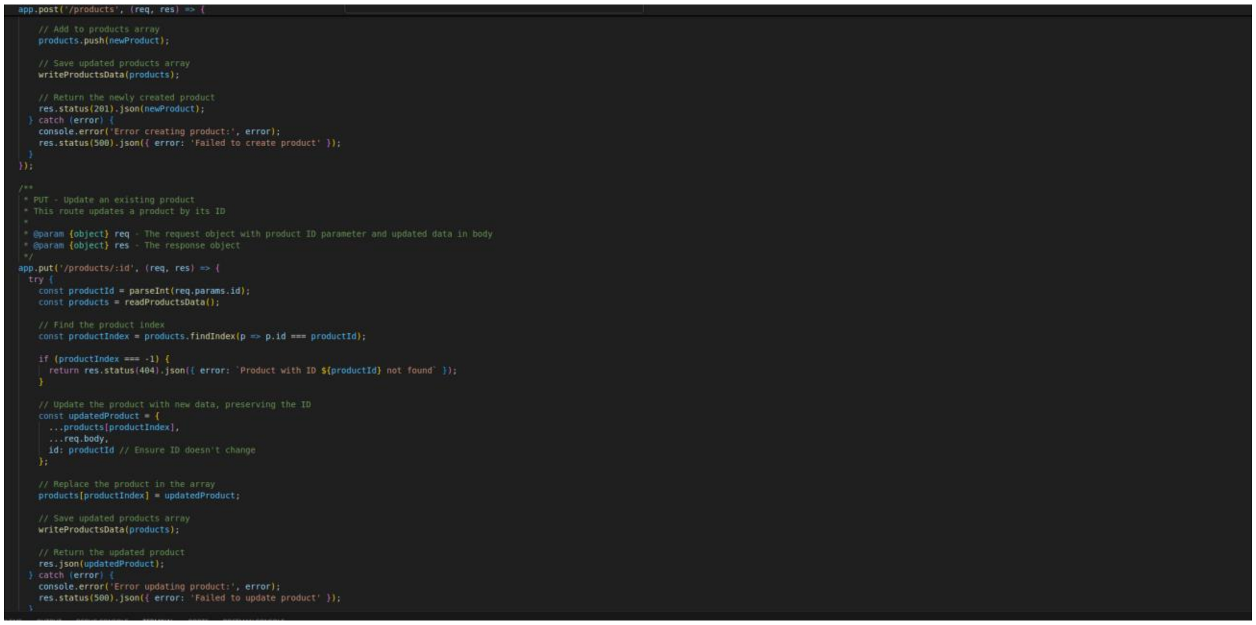

localhost:3000/products

Gmail Maps YouTube Document1.docx (176) Anime Fa... Flight Network

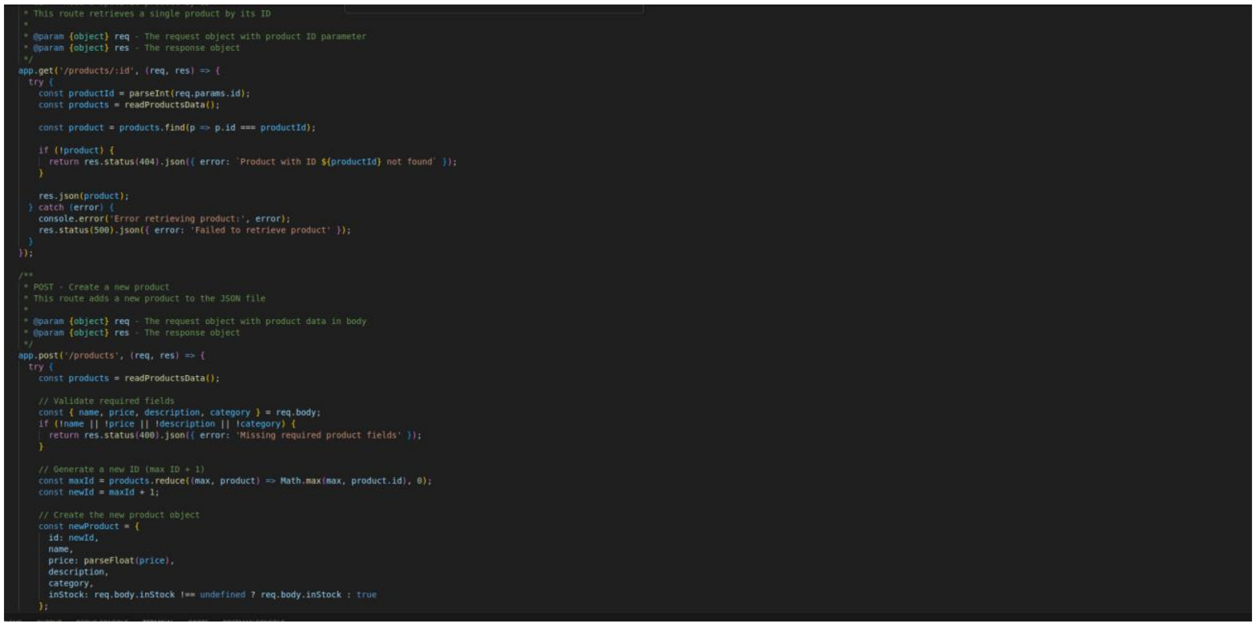
Pretty-print ✓

```
[
  {
    "id": 1,
    "name": "Laptop",
    "price": 999.99,
    "description": "High-performance laptop with 16GB RAM and 512GB SSD",
    "category": "Electronics",
    "inStock": true
  },
  {
    "id": 2,
    "name": "Smartphone",
    "price": 699.99,
    "description": "Latest smartphone with 128GB storage and 5G capability",
    "category": "Electronics",
    "inStock": true
  },
  {
    "id": 3,
    "name": "Headphones",
    "price": 149.99,
    "description": "Noise-cancelling wireless headphones with 30-hour battery life",
    "category": "Audio",
    "inStock": true
  },
  {
    "id": 4,
    "name": "Smartwatch",
    "price": 249.99,
    "description": "Fitness tracker with heart rate monitor and GPS",
    "category": "Wearables",
    "inStock": false
  },
  {
    "id": 5,
    "name": "Tablet",
    "price": 399.99,
    "description": "10-inch tablet with high-resolution display and 64GB storage",
    "category": "Electronics",
    "inStock": true
  },
  {
    "id": 6,
    "name": "Wireless Speaker",
    "price": 79.99,
    "description": "Portable Bluetooth speaker with waterproof design",
    "category": "Audio",
    "inStock": true
  }
]
```

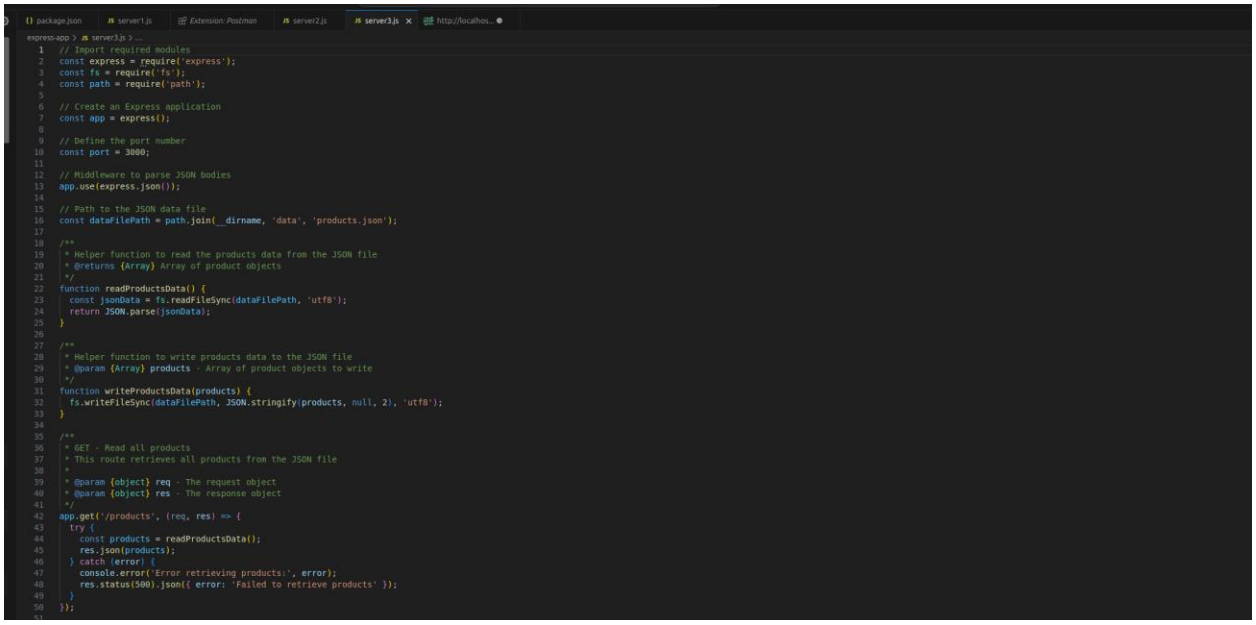
Filecode: 3



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

    // Add to products array
    products.push(newProduct);

    // Save updated products array
    writeProductsData(products);

    // Return the newly created product
    res.status(201).json(newProduct);
  } catch (error) {
    console.error('Error creating product:', error);
    res.status(500).json({ error: 'Failed to create product' });
  }
});

/**
 * PUT - Update an existing product
 * This route updates a product by its ID
 *
 * @param {object} req - The request object with product ID parameter and updated data in body
 * @param {object} res - The response object
 */
app.put('/products/:id', (req, res) => {
  try {
    const productId = parseInt(req.params.id);
    const products = readProductsData();

    // Find the product index
    const productIndex = products.findIndex(p => p.id === productId);

    if (productIndex === -1) {
      return res.status(404).json({ error: `Product with ID ${productId} not found` });
    }

    // Update the product with new data, preserving the ID
    const updatedProduct = {
      ...products[productIndex],
      ...req.body,
      id: productId // Ensure ID doesn't change
    };

    // Replace the product in the array
    products[productIndex] = updatedProduct;

    // Save updated products array
    writeProductsData(products);

    // Return the updated product
    res.json(updatedProduct);
  } catch (error) {
    console.error('Error updating product:', error);
    res.status(500).json({ error: 'Failed to update product' });
  }
```

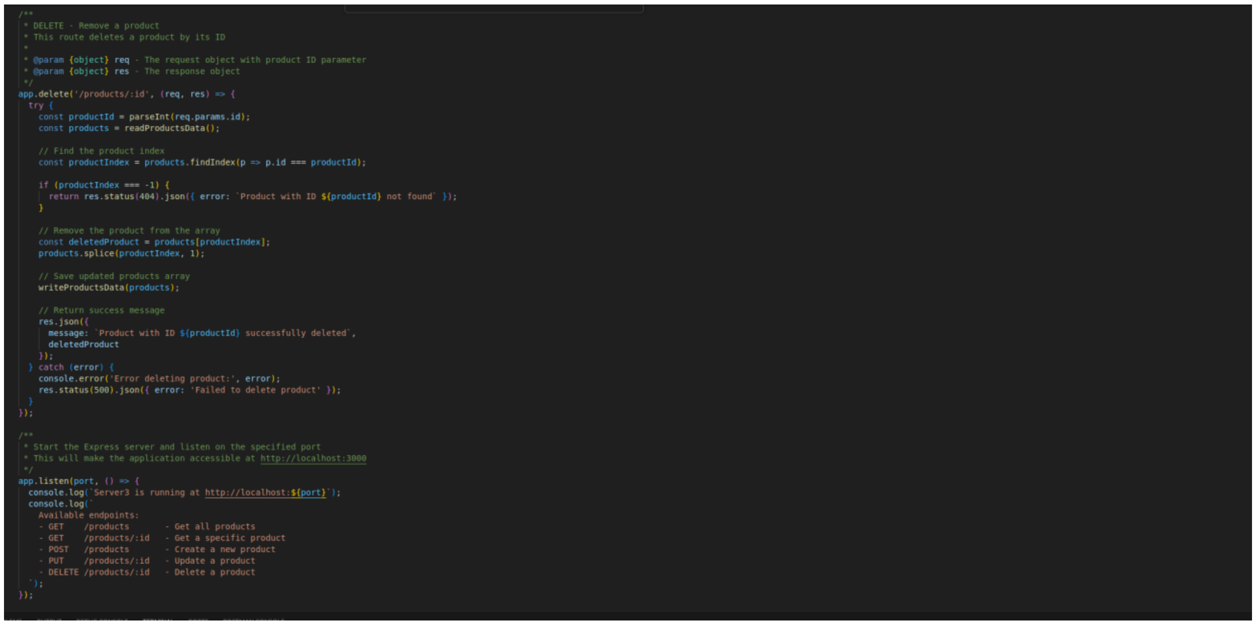


```
 * This route retrieves a single product by its ID
 *
 * @param {object} req - The request object with product ID parameter
 * @param {object} res - The response object
 */
app.get('/products/:id', (req, res) => {
  try {
    const productId = parseInt(req.params.id);
    const products = readProductsData();

    const product = products.find(p => p.id === productId);

    if (!product) {
      return res.status(404).json({ error: `Product with ID ${productId} not found` });
    }

    res.json(product);
  } catch (error) {
    console.error('Error retrieving product:', error);
    res.status(500).json({ error: 'Failed to retrieve product' });
  }
});

/**
 * POST - Create a new product
 * This route adds a new product to the JSON file
 *
 * @param {object} req - The request object with product data in body
 * @param {object} res - The response object
 */
app.post('/products', (req, res) => {
  try {
    const products = readProductsData();

    // Validate required fields
    const { name, price, description, category } = req.body;
    if (!name || !price || !description || !category) {
      return res.status(400).json({ error: 'Missing required product fields' });
    }

    // Generate a new ID (max ID + 1)
    const maxId = products.reduce((max, product) => Math.max(max, product.id), 0);
    const newId = maxId + 1;

    // Create the new product object
    const newProduct = {
      id: newId,
      name,
      price: parseFloat(price),
      description,
      category,
      inStock: req.body.inStock !== undefined ? req.body.inStock : true
    };
```


{} package.json | JS server1.js | Extension: Postman | JS server2.js | JS server3.js | http://localhos...

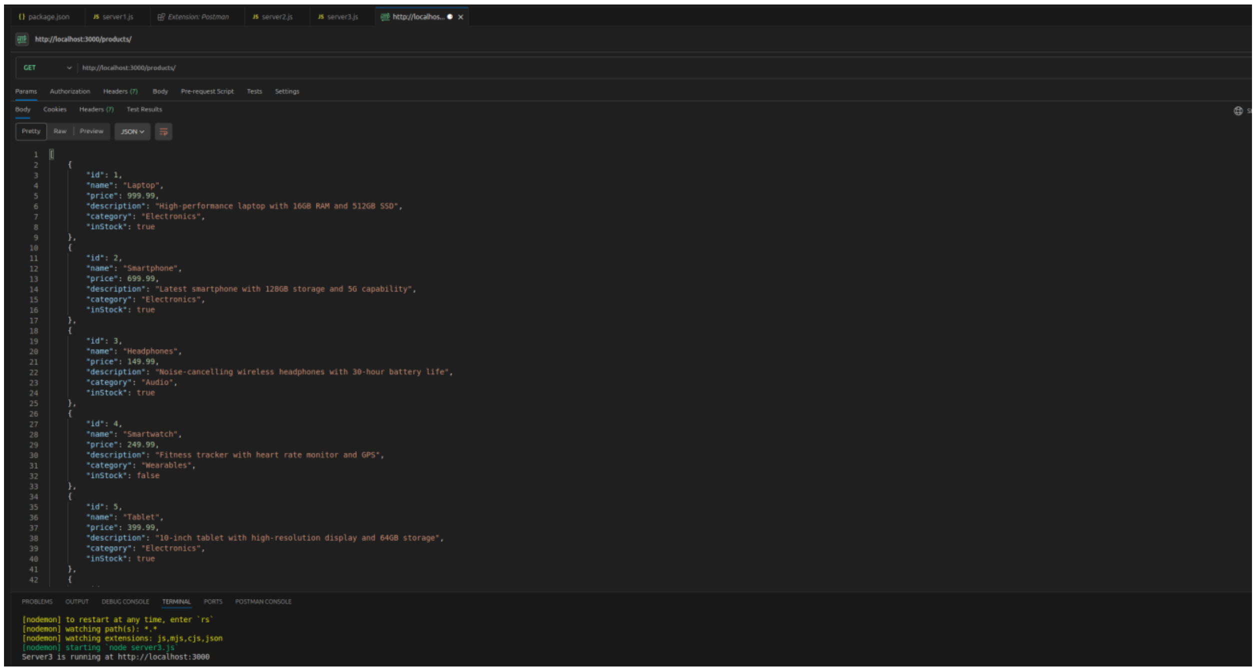
express-app > JS server3.js > ...

```javascript
// Import required modules
const express = require('express');
const fs = require('fs');
const path = require('path');

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

// Define the port number
const port = 3000;

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

// Path to the JSON data file
const dataFilePath = path.join(__dirname, 'data', 'products.json');

/**
 * Helper function to read the products data from the JSON file
 * @returns {Array} Array of product objects
 */
function readProductsData() {
  const jsonData = fs.readFileSync(dataFilePath, 'utf8');
  return JSON.parse(jsonData);
}

/**
 * Helper function to write products data to the JSON file
 * @param {Array} products - Array of product objects to write
 */
function writeProductsData(products) {
  fs.writeFileSync(dataFilePath, JSON.stringify(products, null, 2), 'utf8');
}

/**
 * GET - Read all products
 * This route retrieves all products from the JSON file
 *
 * @param {object} req - The request object
 * @param {object} res - The response object
 */
app.get('/products', (req, res) => {
  try {
    const products = readProductsData();
    res.json(products);
  } catch (error) {
    console.error('Error retrieving products:', error);
    res.status(500).json({ error: 'Failed to retrieve products' });
  }
});
```



```javascript
/**
 * DELETE - Remove a product
 * This route deletes a product by its ID
 *
 * @param {object} req - The request object with product ID parameter
 * @param {object} res - The response object
 */
app.delete('/products/:id', (req, res) => {
  try {
    const productId = parseInt(req.params.id);
    const products = readProductsData();

    // Find the product index
    const productIndex = products.findIndex(p => p.id === productId);

    if (productIndex === -1) {
      return res.status(404).json({ error: `Product with ID ${productId} not found` });
    }

    // Remove the product from the array
    const deletedProduct = products[productIndex];
    products.splice(productIndex, 1);

    // Save updated products array
    writeProductsData(products);

    // Return success message
    res.json({
      message: `Product with ID ${productId} successfully deleted`,
      deletedProduct
    });
  } catch (error) {
    console.error('Error deleting product:', error);
    res.status(500).json({ error: 'Failed to delete product' });
  }
});

/**
 * Start the Express server and listen on the specified port
 * This will make the application accessible at http://localhost:3000
 */
app.listen(port, () => {
  console.log(`Server3 is running at http://localhost:${port}`);
  console.log(`
    Available endpoints:
    - GET    /products       - Get all products
    - GET    /products/:id   - Get a specific product
    - POST   /products       - Create a new product
    - PUT    /products/:id   - Update a product
    - DELETE /products/:id   - Delete a product
  `);
});
```

Response file 3: -



```
GET http://localhost:3000/products/
```

```
[
    {
        "id": 1,
        "name": "Laptop",
        "price": 999.99,
        "description": "High-performance laptop with 16GB RAM and 512GB SSD",
        "category": "Electronics",
        "inStock": true
    },
    {
        "id": 2,
        "name": "Smartphone",
        "price": 699.99,
        "description": "Latest smartphone with 128GB storage and 5G capability",
        "category": "Electronics",
        "inStock": true
    },
    {
        "id": 3,
        "name": "Headphones",
        "price": 149.99,
        "description": "Noise-cancelling wireless headphones with 30-hour battery life",
        "category": "Audio",
        "inStock": true
    },
    {
        "id": 4,
        "name": "Smartwatch",
        "price": 249.99,
        "description": "Fitness tracker with heart rate monitor and GPS",
        "category": "Wearables",
        "inStock": false
    },
    {
        "id": 5,
        "name": "Tablet",
        "price": 399.99,
        "description": "10-inch tablet with high-resolution display and 64GB storage",
        "category": "Electronics",
        "inStock": true
    },
    {
```

```
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server3.js`
Server3 is running at http://localhost:3000
```