

Lab-Assignment#3

JavaScript Frameworks

Jesbin Jobi - 200588911

Rojin Babu – 200587425

Submission Date: 6/19/25

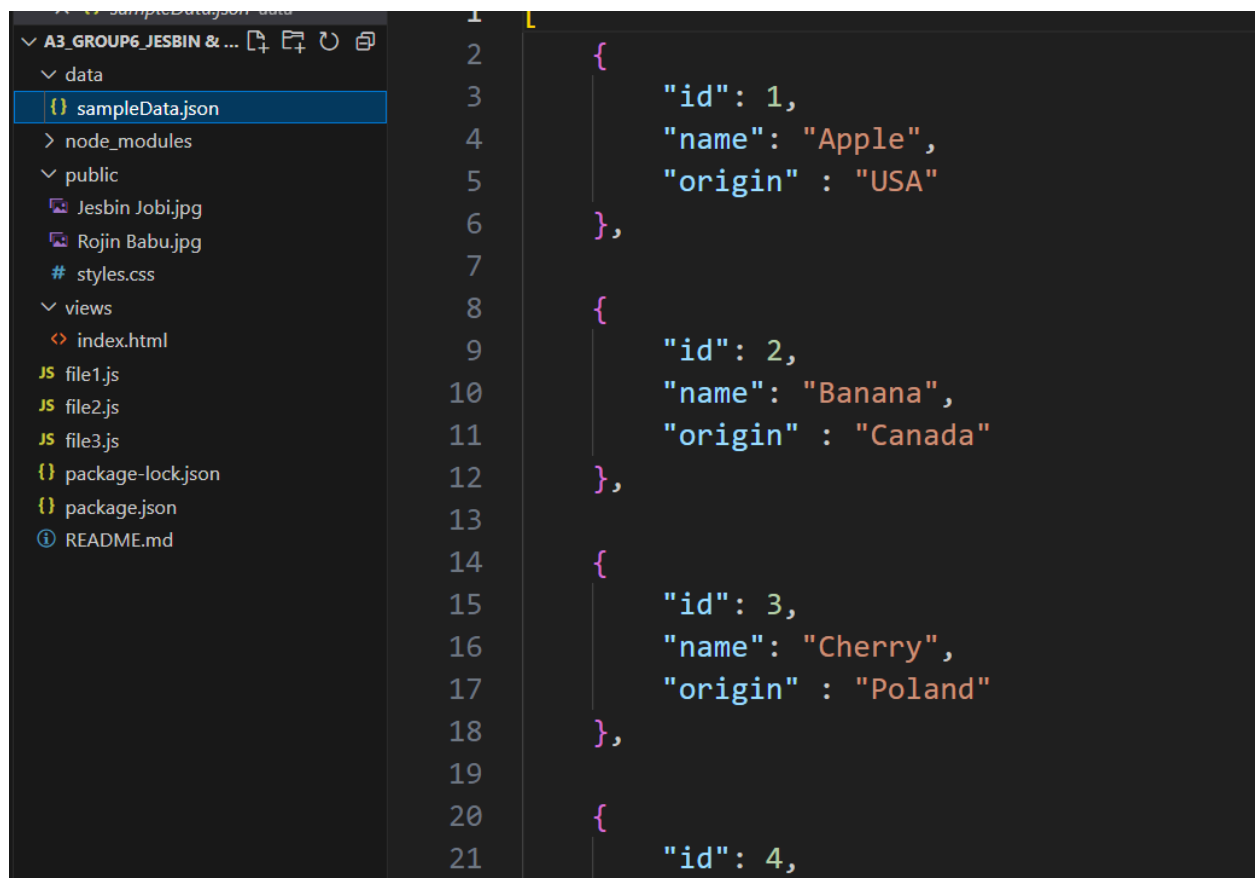
Contents

- **Summary**
- **Screen Shot of folder structure inside Visual Studio Code**
- **Screen Shot of VS Code embedded terminal window**
- **Screen Shot of JavaScript Codes**
- **Code Running Screen Shots**
- **Link to the GitHub Repository**

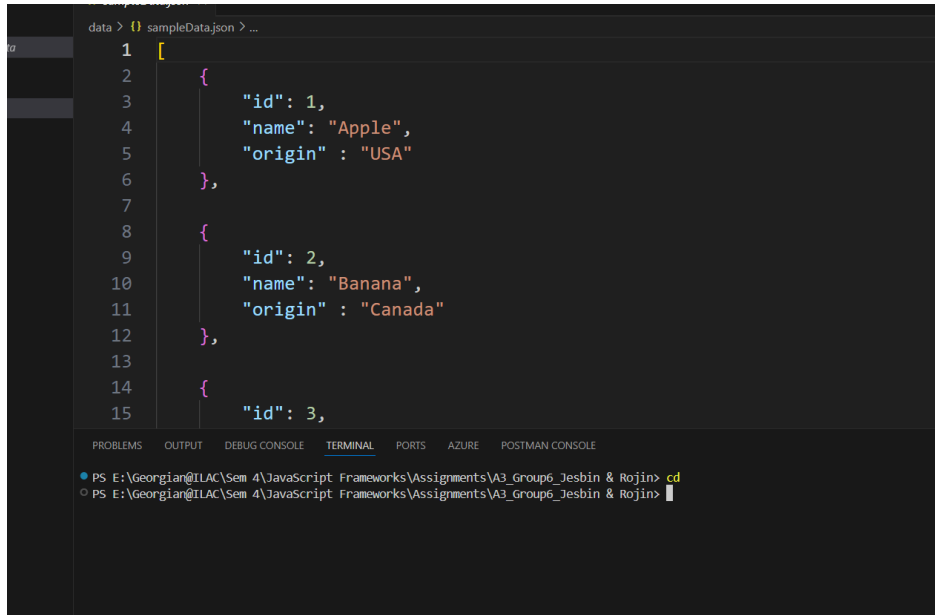
Summary Of the Project

in this assignment, we have had an opportunity to work with simple Express.js server creation and implement CRUD (Create, Read, Update, Delete) functionality with JSON data. We developed various API endpoints to show, create, modify, and remove data about fruit. To make the project more realistic, we also learned how to serve static files like CSS and images, and how to properly organize our project folders.

Screen Shot of folder structure inside Visual Studio Code



Screen Shot of VS Code embedded terminal window



The screenshot shows the VS Code editor interface. The main editor area displays a JSON file named `sampleData.json` with the following content:

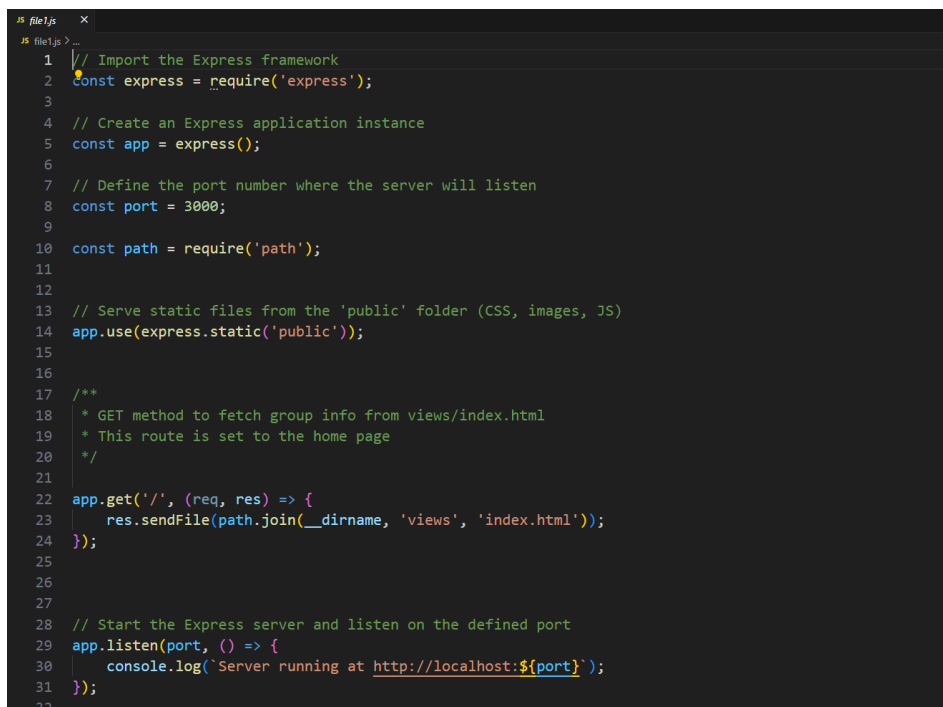
```
1  [
2    {
3      "id": 1,
4      "name": "Apple",
5      "origin": "USA"
6    },
7
8    {
9      "id": 2,
10     "name": "Banana",
11     "origin": "Canada"
12   },
13
14   {
15     "id": 3,
```

The bottom panel of VS Code shows the **TERMINAL** tab. It contains two command prompt sessions:

```
PS E:\Georgian@ILAC\Sem 4\JavaScript Frameworks\Assignments\A3_Group6_Jesbin & Rojin> cd
PS E:\Georgian@ILAC\Sem 4\JavaScript Frameworks\Assignments\A3_Group6_Jesbin & Rojin>
```

Screen Shot of JavaScript Codes

File.js



The screenshot shows the VS Code editor with a file named `File.js`. The code is as follows:

```
1  // Import the Express framework
2  const express = require('express');
3
4  // Create an Express application instance
5  const app = express();
6
7  // Define the port number where the server will listen
8  const port = 3000;
9
10 const path = require('path');
11
12
13 // Serve static files from the 'public' folder (CSS, images, JS)
14 app.use(express.static('public'));
15
16
17 /**
18  * GET method to fetch group info from views/index.html
19  * This route is set to the home page
20  */
21
22 app.get('/', (req, res) => {
23   res.sendFile(path.join(__dirname, 'views', 'index.html'));
24 });
25
26
27
28 // Start the Express server and listen on the defined port
29 app.listen(port, () => {
30   console.log(`Server running at http://localhost:${port}`);
31 });
32
```

File2.js

```
JS file2.js x
JS file2.js > ...
1 // Import Express and Framework for creating the server
2 const express = require('express');
3
4 // creating an instance of Express app
5 const app = express();
6
7 // defining the port number
8 const port = 3000;
9
10 // Importing Nodes.js file system module to read
11 const fs = require('fs');
12
13 // Define a GET route to handle requests to the '/fruits' URL path
14 app.get('/fruits', (req, res) => {
15   // Reading the json file from data folder
16   fs.readFile('./data/sampleData.json', 'utf8', (err, data) => {
17     // if any error, it will stop the execution
18     if (err) throw err;
19     res.json(JSON.parse(data)); // Displays raw JSON data
20   });
21 });
22
23 // Start the server
24 app.listen(port, () => {
25   console.log(`Server running at http://localhost:${port}`);
26 });
27
```

File3.js

```
JS file3.js x
JS file3.js > ...
1 const express = require('express');
2 const app = express();
3 const port = 3000;
4
5 // To handle JSON data from Postman
6 app.use(express.json());
7
8 // Load data from JSON file
9 let fruits = require('./data/sampleData.json');
10
11 // Displaying a welcome message in the root URL
12 app.get('/', (req, res) => {
13   res.send('<h1>Welcome to CRUD Operations</h1><p>Use Postman to test CRUD routes.</p>');
14 });
15
16 // READ - Get all fruits
17 // Method: GET
18 // Sending the list of fruit as a Json
19
20 app.get('/fruits', (req, res) => {
21   res.json(fruits);
22 });
23
24 // CREATE - Add a new fruit
25 // Method: POST
26 // Sending the new fruit as a Json
27
28 app.post('/fruits', (req, res) => {
29   const newFruit = req.body; // Getting the new fruit data from req.body
30
```

```

JS file3.js x
JS file3.js > ...
28 app.post('/fruits', (req, res) => {
29
30     // Validate incoming data with the same ID
31     if (!newFruit || !newFruit.id || !newFruit.name) {
32         return res.status(400).json({ message: 'Invalid fruit data. Please provide id and name.' });
33     }
34
35     // Check if fruit with same id already exists
36     const exists = fruits.find(fruit => fruit.id === newFruit.id);
37     if (exists) {
38         return res.status(400).json({ message: 'Fruit with this ID already exists' });
39     }
40
41     fruits.push(newFruit);
42     res.json({ message: 'Fruit added successfully', data: newFruit });
43 });
44
45 // UPDATE - Edit a fruit by id
46 // Method: PUT
47 // Sending the updated fruit as a Json
48
49
50 app.put('/fruits/:id', (req, res) => {
51     const id = parseInt(req.params.id);
52     const updatedFruit = req.body;
53
54     if (!updatedFruit || !updatedFruit.name) {
55         return res.status(400).json({ message: 'Invalid data. Please provide name.' });
56     }
57
58     const fruitIndex = fruits.findIndex(fruit => fruit.id === id);
59

```

```

JS file3.js x
JS file3.js > ...
50 app.put('/fruits/:id', (req, res) => {
62     res.json({ message: 'Fruit updated successfully', data: fruits[fruitIndex] });
63 } else {
64     res.status(404).json({ message: 'Fruit not found' });
65 }
66 });
67
68 // DELETE - Remove a fruit by id
69 // Method: DELETE
70 // Sending the id of the fruit to be deleted as a Json
71
72 app.delete('/fruits/:id', (req, res) => {
73     const id = parseInt(req.params.id); // Extract ID from URL
74
75     // Find the index of the fruit with the matching ID
76     const fruitIndex = fruits.findIndex(fruit => fruit.id === id);
77
78     if (fruitIndex !== -1) {
79         const removedFruit = fruits.splice(fruitIndex, 1); // Removing the fruit by ID
80         res.json({ message: 'Fruit deleted successfully', data: removedFruit });
81     } else {
82         // If not found, returning error
83         res.status(404).json({ message: 'Fruit not found' });
84     }
85 });
86
87 // Start the server
88 app.listen(port, () => {
89     console.log(`Server running at http://localhost:${port}`);
90 });
91

```

Index and CSS File

```

index.html
views > index.html > html > body > div.card-container
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Group Number 6</title>
7   <link rel="stylesheet" href="/styles.css" /> <!-- It will look for css file in root of the public folder by Ex
8 </head>
9 <body>
10  <h1>Lab-Assignment#3</h1>
11  <h2>Group 6</h2>
12
13  <div class="card-container">
14
15    <div class="profile-card">
16       <!-- It will look for img file in root of the public
17      <h2>Jesbin Jobi</h2>
18      <p>ID: 200588911</p>
19      <p>JavaScript Frameworks</p>
20    </div>/.profile-card
21
22    <div class="profile-card">
23      
24      <h2>Rojin Babu</h2>
25      <p>ID: 200587425</p>
26      <p>JavaScript Frameworks</p>
27    </div>/.profile-card
28
29  </div>/.card-container
30
31 </body>
32 </html>

```

```

styles.css
public > styles.css
1 @import url('https://fonts.googleapis.com/css2?family=SpaceGrotesk:wght@400;500;600;700&display=swap');
2
3 body{
4   background-color: #f0f0f0;
5   font-family: 'Space Grotesk', sans-serif;
6   display: grid;
7   place-items: center;
8 }
9
10
11 h1{
12   font-size: 45px;
13   font-weight: 800;
14   color: #ffffff;
15 }
16
17 h2{
18   font-size: 40px;
19   font-weight: 800;
20 }
21
22 .card-container {
23   display: flex;
24   justify-content: center;
25   gap: 20px;
26 }
27
28 .profile-card {
29   background-color: #ffffff;
30   border: 1px solid #721212;
31   border-radius: 15px;
32   width: 250px;
33   padding: 20px;
34   box-shadow: 0 4px 8px #0000000a;
35   transition: transform 0.3s;
36 }
37
38 .profile-card:hover {
39   transform: scale(1.05);
40 }
41
42 .profile-card img {
43   align-items: center;
44   width: 150px;
45   height: 150px;
46   margin-left: 43px;
47 }
48
49 .profile-card h2 {
50   color: #40566c;
51   margin: 15px 0 5px 0;
52 }
53
54 .profile-card p {
55   color: #6d5628;
56   margin: 5px 0;
57 }

```

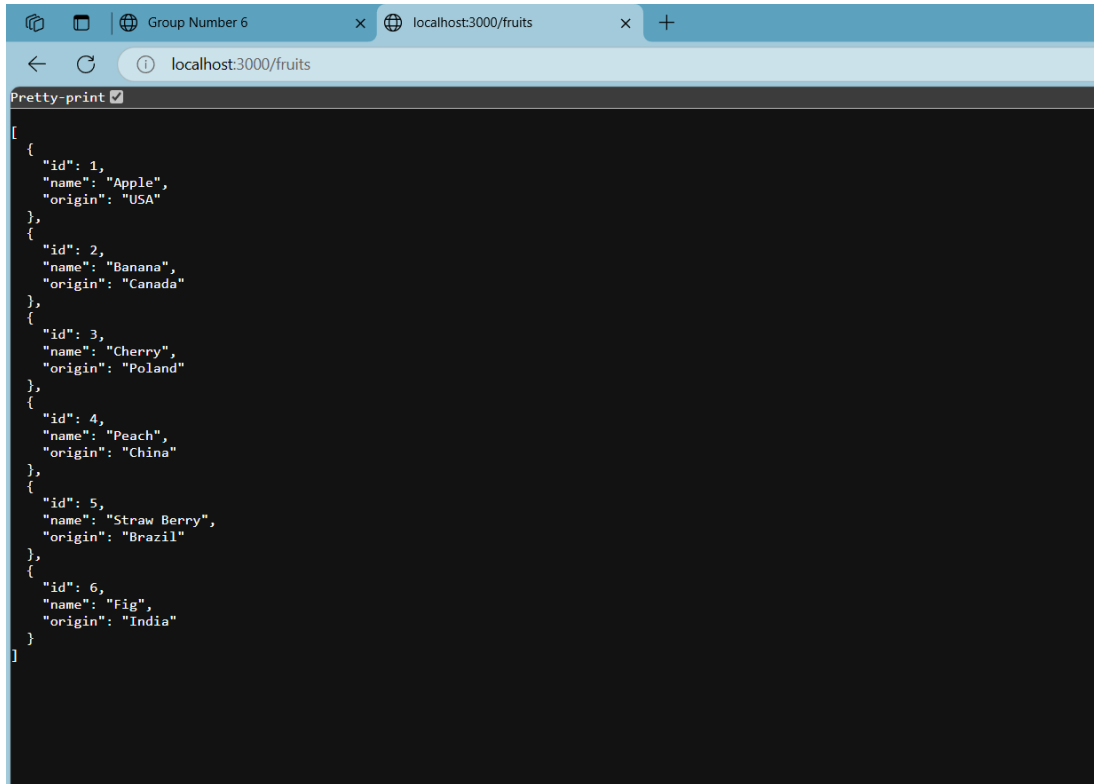
Nodemon

```
PS E:\Georgian@ILAC\Sem 4\JavaScript Frameworks\Assignments\A3_Group6_Jesbin & Rojin> nodemon file1.js
>>
[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node file1.js`
Server running at http://localhost:3000
PS E:\Georgian@ILAC\Sem 4\JavaScript Frameworks\Assignments\A3_Group6_Jesbin & Rojin> nodemon file2.js
>>
[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node file2.js`
Server running at http://localhost:3000
PS E:\Georgian@ILAC\Sem 4\JavaScript Frameworks\Assignments\A3_Group6_Jesbin & Rojin> nodemon file3.js
>>
[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node file3.js`
Server running at http://localhost:3000
```

File1.js



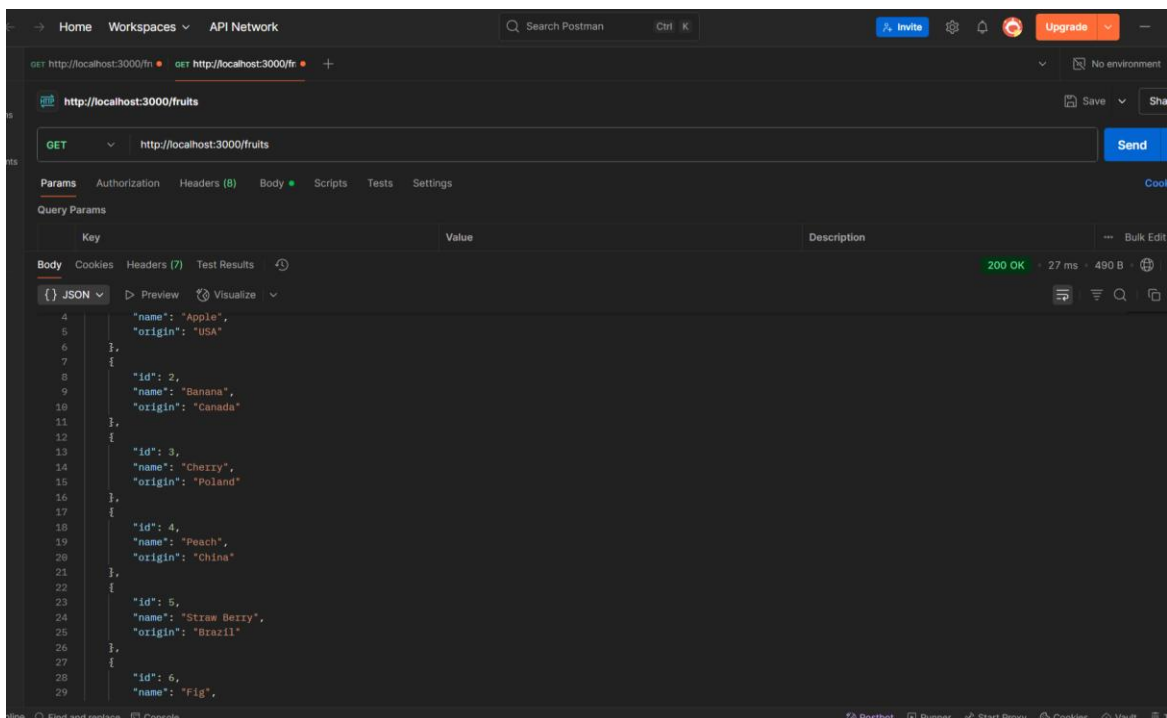
File2.js



A screenshot of a web browser window. The address bar shows 'localhost:3000/fruits'. The page content is a JSON array of six fruit objects, displayed with 'Pretty-print' checked. The data is as follows:

```
[
  {
    "id": 1,
    "name": "Apple",
    "origin": "USA"
  },
  {
    "id": 2,
    "name": "Banana",
    "origin": "Canada"
  },
  {
    "id": 3,
    "name": "Cherry",
    "origin": "Poland"
  },
  {
    "id": 4,
    "name": "Peach",
    "origin": "China"
  },
  {
    "id": 5,
    "name": "Straw Berry",
    "origin": "Brazil"
  },
  {
    "id": 6,
    "name": "Fig",
    "origin": "India"
  }
]
```

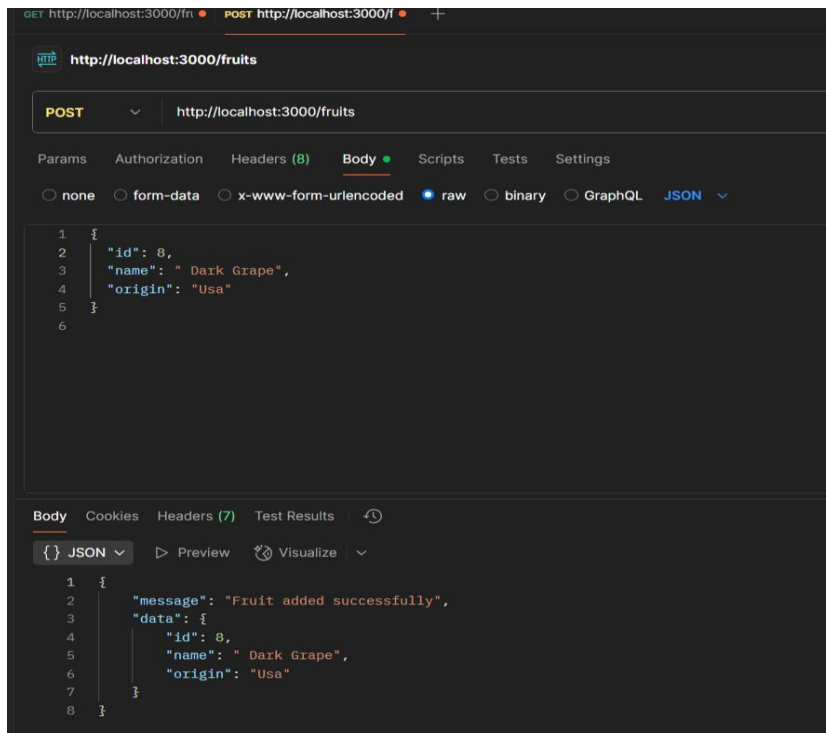
File3.js in Post man



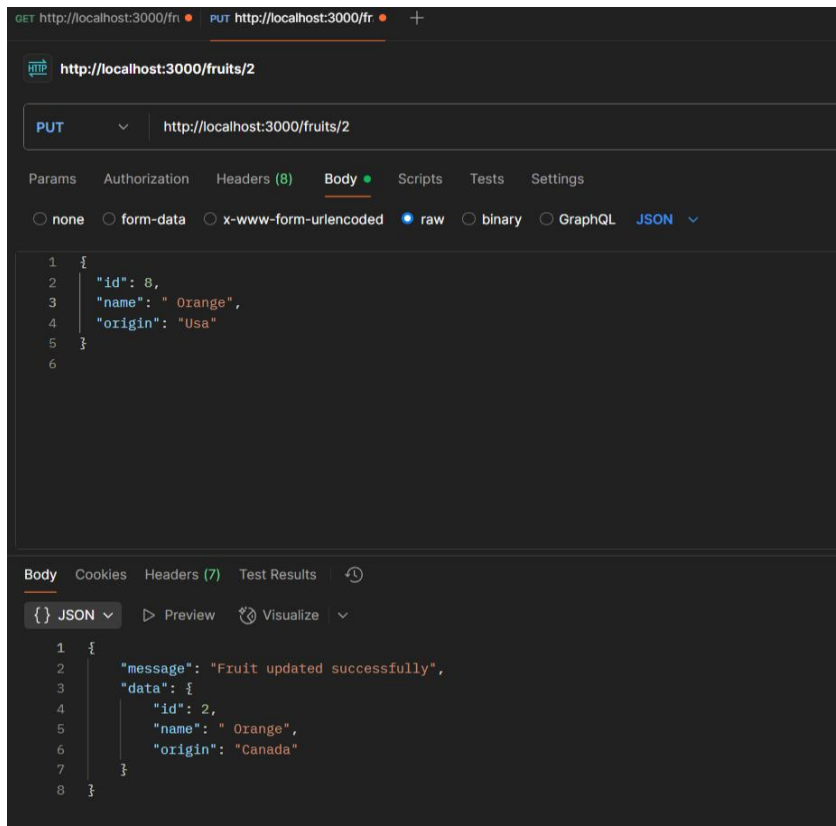
A screenshot of the Postman application. A GET request to 'http://localhost:3000/fruits' has been sent. The response is a JSON array of six fruit objects, displayed in the 'Body' tab. The status is '200 OK' with a response time of 27 ms and a size of 490 B. The JSON data is as follows:

```
[
  {
    "name": "Apple",
    "origin": "USA"
  },
  {
    "id": 2,
    "name": "Banana",
    "origin": "Canada"
  },
  {
    "id": 3,
    "name": "Cherry",
    "origin": "Poland"
  },
  {
    "id": 4,
    "name": "Peach",
    "origin": "China"
  },
  {
    "id": 5,
    "name": "Straw Berry",
    "origin": "Brazil"
  },
  {
    "id": 6,
    "name": "Fig",
    "origin": "India"
  }
]
```

POST method



PUT method



Link to the GitHub Repository - [git@github.com:Jesbin-Github/Lab-Assignment-3.git](https://github.com/Jesbin-Github/Lab-Assignment-3.git)