

Practical No. 7

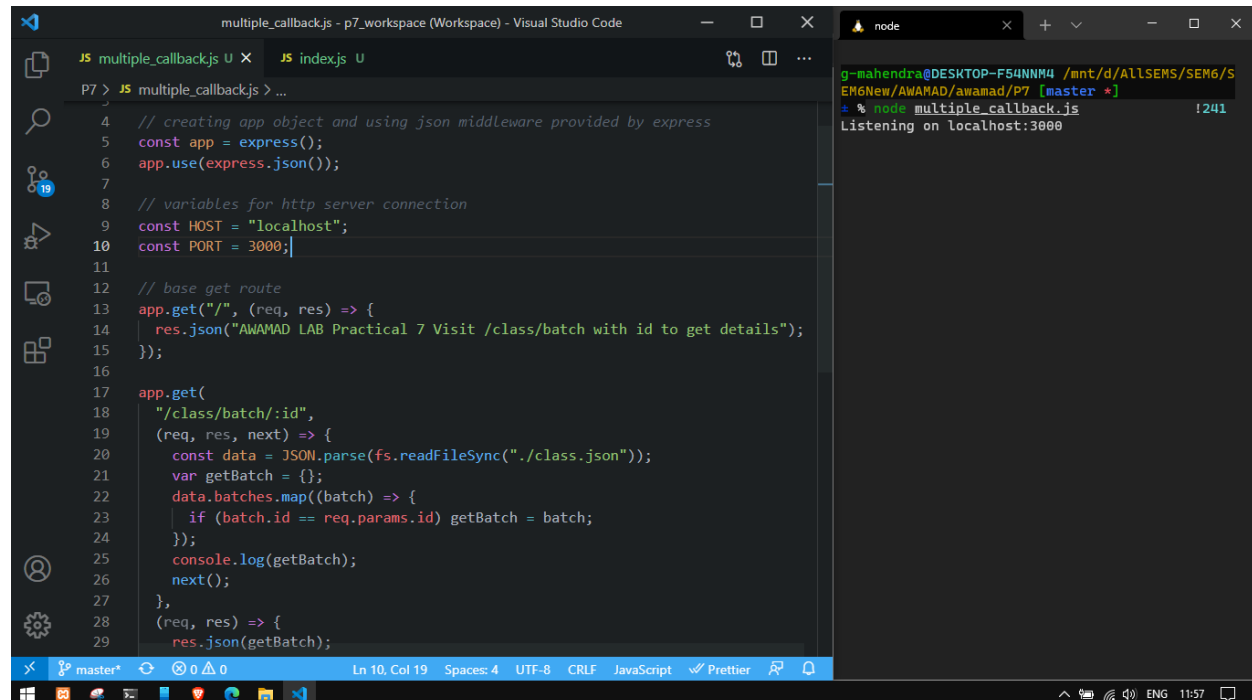
Exam Seat No:

1. 2018BTECS00033- Mahendra Bhimrao Gharge
2. GitHub URL: <https://github.com/g-mahendra/awamad>

Problem Statement 1:

1. Write an Express app that uses the multiple call-back functions to handle a request.
2. Use different routes to show the use of call-back functions.
3. Use at least two call-back functions.
4. Use array of functions as an argument.

Screenshot 1:



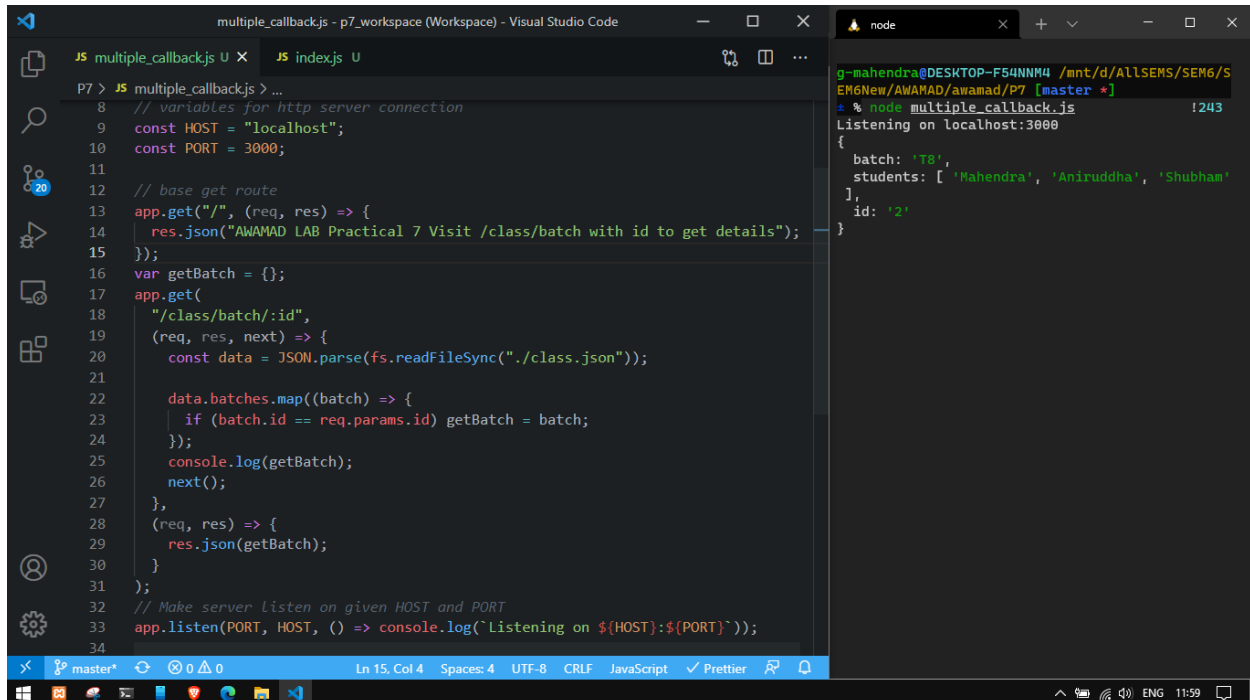
The screenshot displays the Visual Studio Code editor with a workspace named 'p7_workspace'. The file 'multiple_callbacks.js' is open, showing the following code:

```
4 // creating app object and using json middleware provided by express
5 const app = express();
6 app.use(express.json());
7
8 // variables for http server connection
9 const HOST = "localhost";
10 const PORT = 3000;
11
12 // base get route
13 app.get("/", (req, res) => {
14   res.json("AWAMAD LAB Practical 7 Visit /class/batch with id to get details");
15 });
16
17 app.get(
18   "/class/batch/:id",
19   (req, res, next) => {
20     const data = JSON.parse(fs.readFileSync("./class.json"));
21     var getBatch = {};
22     data.batches.map((batch) => {
23       if (batch.id == req.params.id) getBatch = batch;
24     });
25     console.log(getBatch);
26     next();
27   },
28   (req, res) => {
29     res.json(getBatch);
30   }
31 );
```

The terminal window on the right shows the command `node multiple_callbacks.js` being executed, resulting in the message: `Listening on localhost:3000`.

Information 1: Installed express with `npm i express`

Screenshot 2:

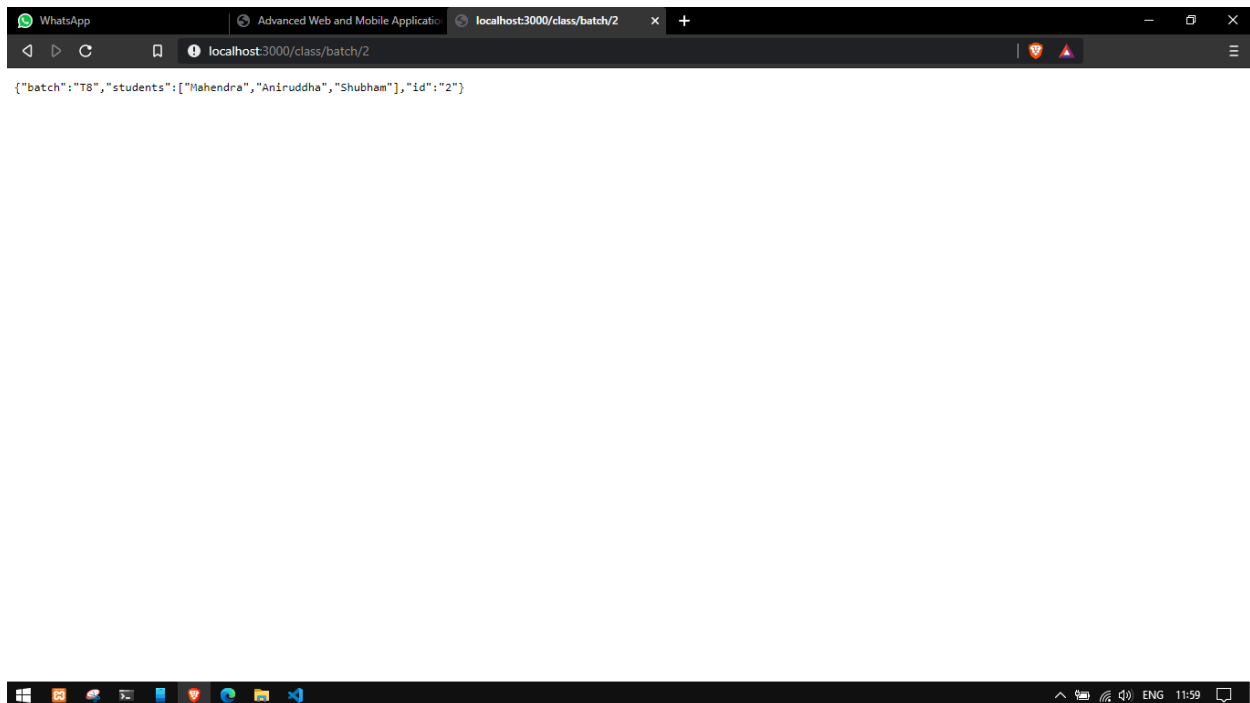


```
multiple_callbackjs U X JS indexjs U
P7 > JS multiple_callbackjs > ...
8 // variables for http server connection
9 const HOST = "localhost";
10 const PORT = 3000;
11
12 // base get route
13 app.get("/", (req, res) => {
14   res.json("AWAMAD LAB Practical 7 Visit /class/batch with id to get details");
15 });
16 var getBatch = {};
17 app.get(
18   "/class/batch/:id",
19   (req, res, next) => {
20     const data = JSON.parse(fs.readFileSync("./class.json"));
21
22     data.batches.map((batch) => {
23       if (batch.id == req.params.id) getBatch = batch;
24     });
25     console.log(getBatch);
26     next();
27   },
28   (req, res) => {
29     res.json(getBatch);
30   }
31 );
32 // Make server listen on given HOST and PORT
33 app.listen(PORT, HOST, () => console.log(`Listening on ${HOST}:${PORT}`));
34
```

```
g-mahendra@DESKTOP-F54NNM4 /mnt/d/ALLSEMS/SEM6/S
EM6New/AWAMAD/awamad/P7 [master *]
+ % node multiple_callback.js 1243
Listening on localhost:3000
{
  batch: 'T8',
  students: [ 'Mahendra', 'Aniruddha', 'Shubham'
],
  id: '2'
}
```

Information 2: Using multiple callbacks for retrieving and sending JSON data.

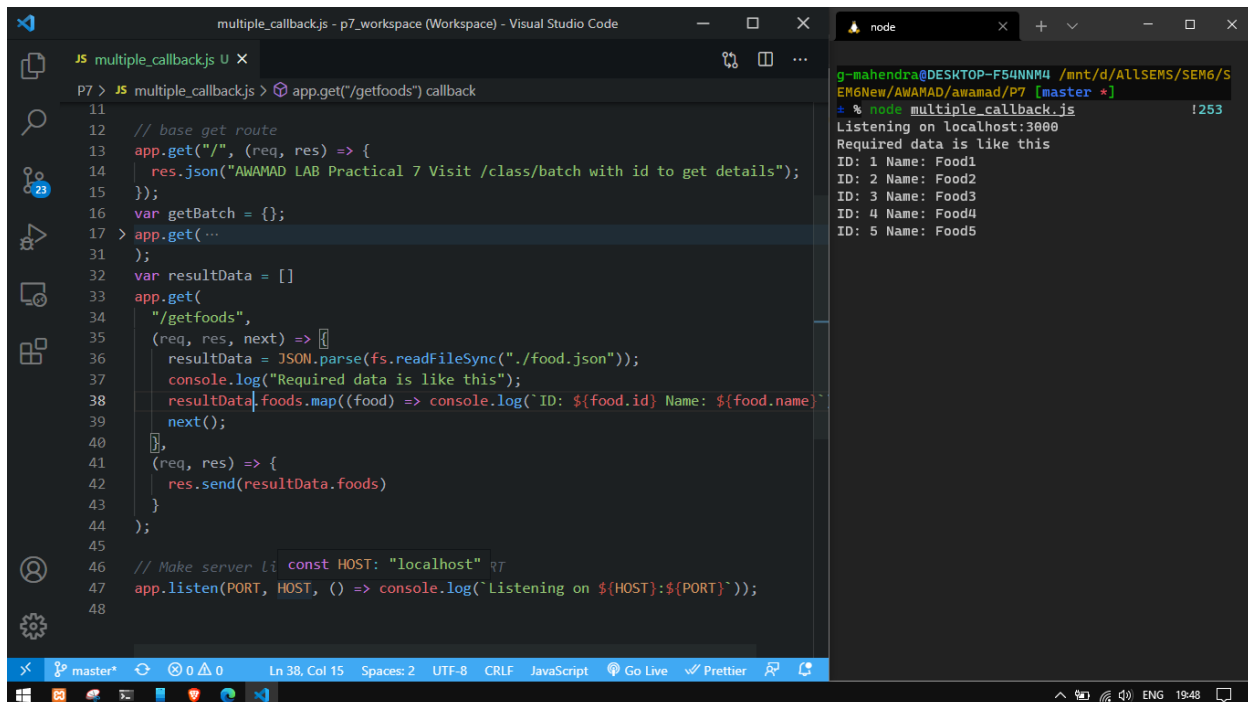
Screenshot 3:



```
localhost:3000/class/batch/2
{"batch": "T8", "students": ["Mahendra", "Aniruddha", "Shubham"], "id": "2"}
```

Information 3: Output of data sent using multiple callbacks

Screenshot 4:



The screenshot shows a Visual Studio Code editor with a file named `multiple_callback.js`. The code defines a base get route and a specific route for `/getfoods`. The `/getfoods` route uses two callback functions: one to read and parse the `food.json` file, and another to send the parsed data as a JSON response. The terminal output shows the server listening on `localhost:3000` and the required data being logged.

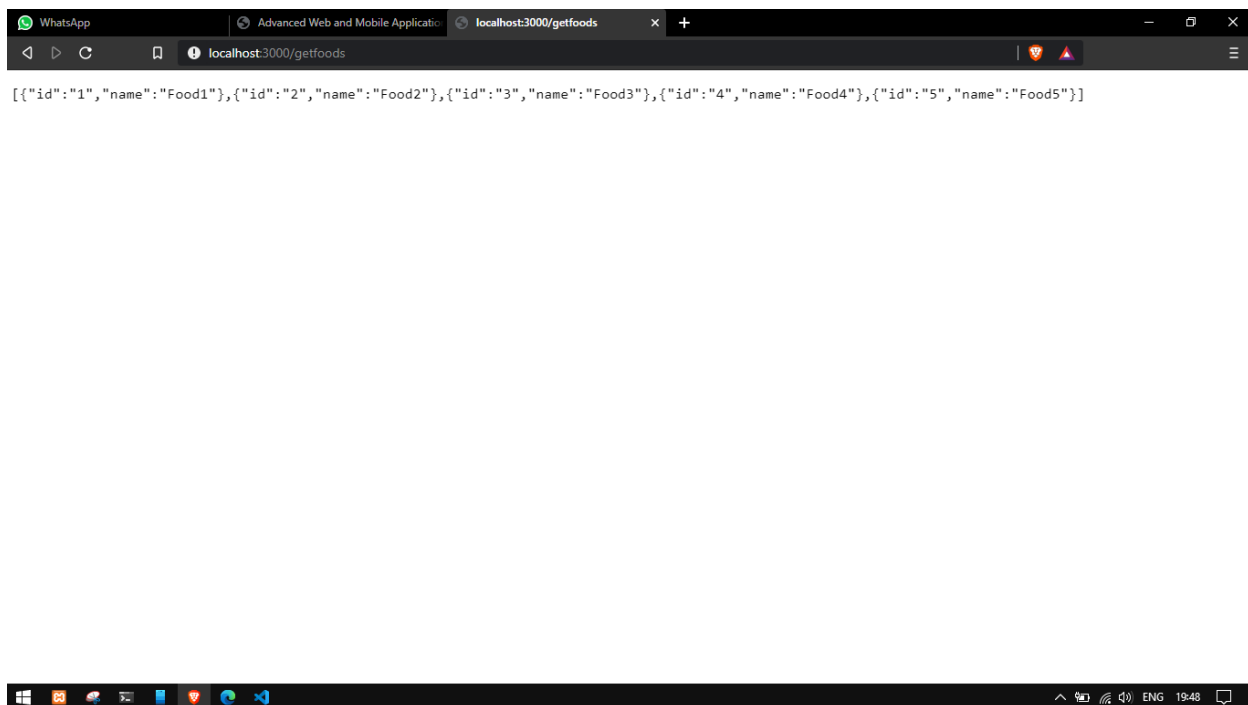
```
11 P7 > JS multiple_callbackjs > app.get("/getfoods") callback
12 // base get route
13 app.get("/", (req, res) => {
14   res.json("AWAMAD LAB Practical 7 Visit /class/batch with id to get details");
15 });
16 var getBatch = {};
17 > app.get(...)
31 );
32 var resultData = []
33 app.get(
34   "/getfoods",
35   (req, res, next) => {
36     resultData = JSON.parse(fs.readFileSync("./food.json"));
37     console.log("Required data is like this");
38     resultData.foods.map((food) => console.log(`ID: ${food.id} Name: ${food.name}`));
39     next();
40   },
41   (req, res) => {
42     res.send(resultData.foods)
43   }
44 );
45
46 // Make server li const HOST: "localhost" 87
47 app.listen(PORT, HOST, () => console.log(`Listening on ${HOST}:${PORT}`));
48
```

Terminal Output:

```
g-mahendra@DESKTOP-F54NNM4 /mnt/d/ALLSEMS/SEM6/S
EM6New/AWAMAD/awamad/P7 [master *]
+ % node multiple_callback.js 1253
Listening on localhost:3000
Required data is like this
ID: 1 Name: Food1
ID: 2 Name: Food2
ID: 3 Name: Food3
ID: 4 Name: Food4
ID: 5 Name: Food5
```

Information 4: Using route `/getfoods` to send data of foods using two call back functions

Screenshot 5:

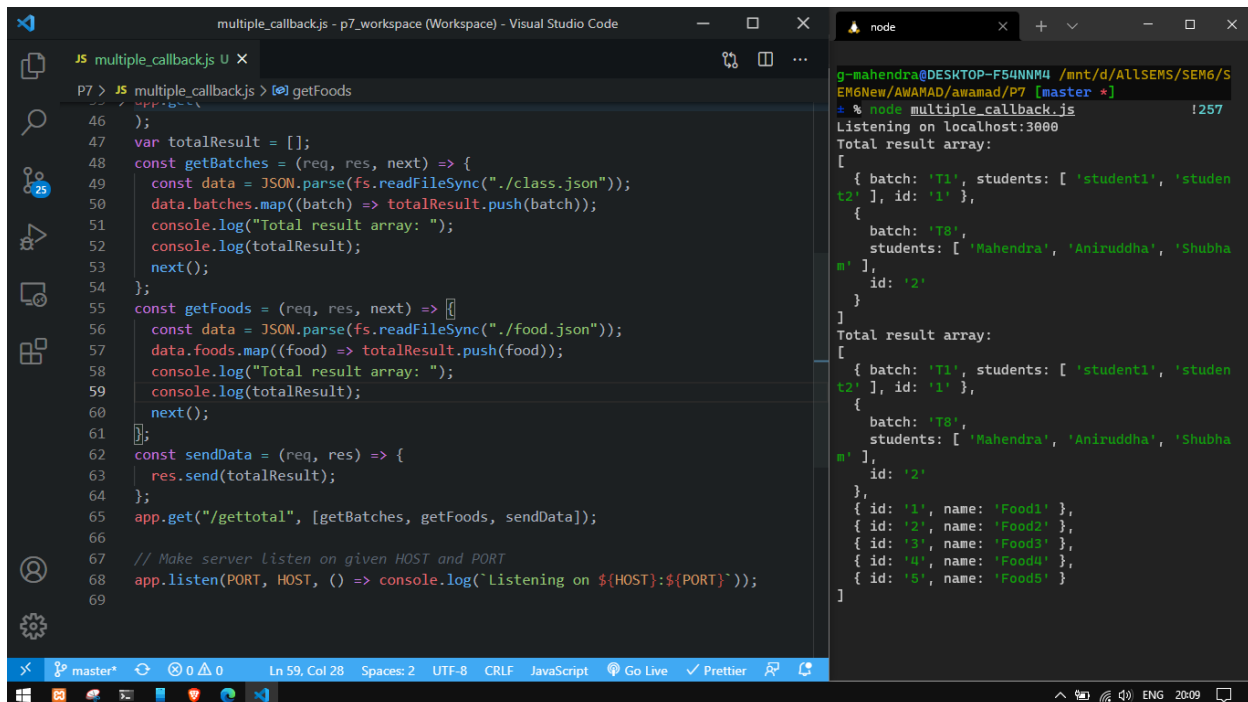


The screenshot shows a web browser window with the address `localhost:3000/getfoods`. The output is a JSON array of food objects, each with an `id` and a `name` property.

```
[{"id": "1", "name": "Food1"}, {"id": "2", "name": "Food2"}, {"id": "3", "name": "Food3"}, {"id": "4", "name": "Food4"}, {"id": "5", "name": "Food5"}]
```

Information 5: Output of `getFoods` route

Screenshot 6:



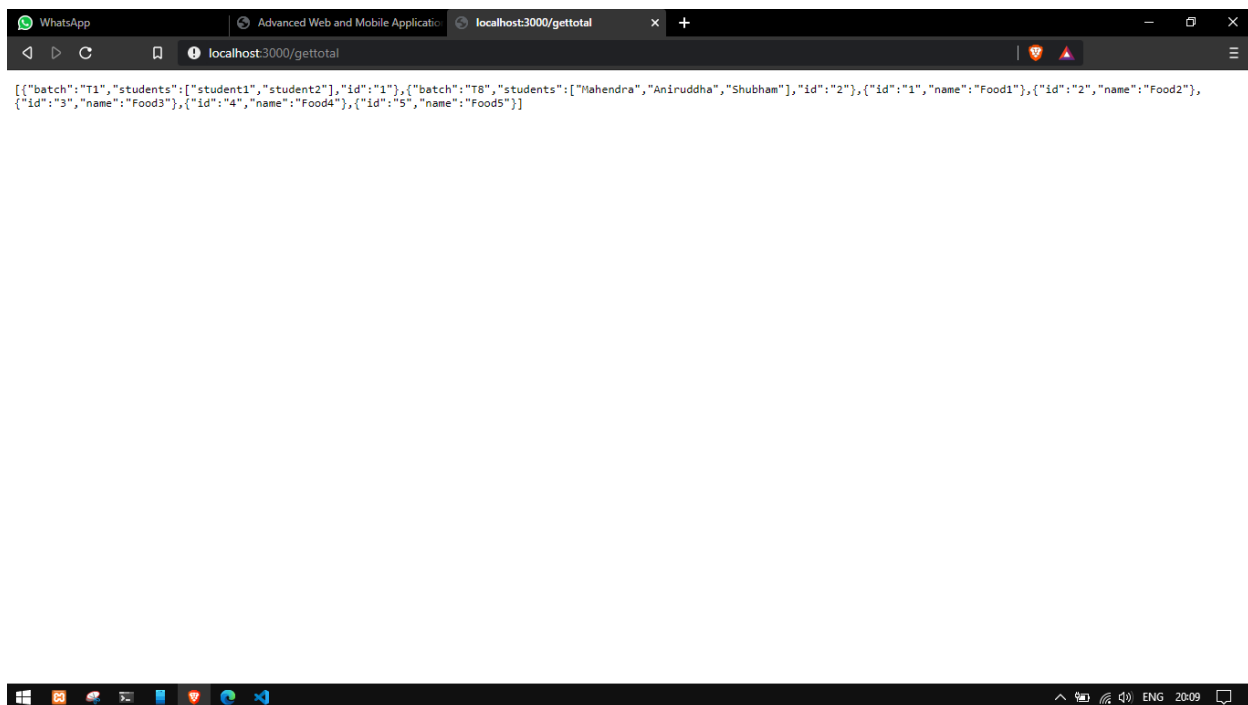
The screenshot shows a Visual Studio Code editor with a file named `multiple_callback.js`. The code defines three asynchronous functions: `getBatches`, `getFoods`, and `sendData`. `getBatches` reads `./class.json`, `getFoods` reads `./food.json`, and `sendData` sends the combined results. A REST client GET request to `/gettotal` triggers these functions. The terminal output shows the console logs and the JSON response from the server.

```
46 );
47 var totalResult = [];
48 const getBatches = (req, res, next) => {
49   const data = JSON.parse(fs.readFileSync("./class.json"));
50   data.batches.map((batch) => totalResult.push(batch));
51   console.log("Total result array: ");
52   console.log(totalResult);
53   next();
54 };
55 const getFoods = (req, res, next) => {
56   const data = JSON.parse(fs.readFileSync("./food.json"));
57   data.foods.map((food) => totalResult.push(food));
58   console.log("Total result array: ");
59   console.log(totalResult);
60   next();
61 };
62 const sendData = (req, res) => {
63   res.send(totalResult);
64 };
65 app.get("/gettotal", [getBatches, getFoods, sendData]);
66
67 // Make server listen on given HOST and PORT
68 app.listen(PORT, HOST, () => console.log(`Listening on ${HOST}:${PORT}`));
69
```

```
g-mahendra@DESKTOP-F54NNM4 /mnt/d/ALLSEMS/SEM6/SEM6New/AMAMAD/awamad/P7 [master *]
+ % node multiple_callback.js 1257
Listening on localhost:3000
Total result array:
[
  { batch: 'T1', students: [ 'student1', 'student2' ], id: '1' },
  { batch: 'T8', students: [ 'Mahendra', 'Aniruddha', 'Shubham' ], id: '2' }
]
Total result array:
[
  { batch: 'T1', students: [ 'student1', 'student2' ], id: '1' },
  { batch: 'T8', students: [ 'Mahendra', 'Aniruddha', 'Shubham' ], id: '2' },
  { id: '1', name: 'Food1' },
  { id: '2', name: 'Food2' },
  { id: '3', name: 'Food3' },
  { id: '4', name: 'Food4' },
  { id: '5', name: 'Food5' }
]
```

Information 6: Using three callbacks with array of callbacks

Screenshot 7:



The screenshot shows a web browser window with the address `localhost:3000/gettotal`. The page displays the JSON response from the REST client GET request, which is the same output shown in the terminal of Screenshot 6.

```
[{"batch": "T1", "students": ["student1", "student2"], "id": "1"}, {"batch": "T8", "students": ["Mahendra", "Aniruddha", "Shubham"], "id": "2"}, {"id": "1", "name": "Food1"}, {"id": "2", "name": "Food2"}, {"id": "3", "name": "Food3"}, {"id": "4", "name": "Food4"}, {"id": "5", "name": "Food5"}]
```

Information 7: output of using three callbacks with array of callbacks

