Learner Assignment Submission Format

Learner Details

- **Name:H O Akash**
- **Enrollment Number:SU625MR015**
- **Batch / Class:MERN Stack**
- **Assignment:Express JS**
- **Date of Submission:13/08/2025**

Problem Solving Activity 1.1

1. Program Statement

To design and implement a Food Ordering Application backend that provides a REST API for basic CRUD (Create, Read, Update, Delete) operations.

---

2. Algorithm

```
Algorithm: Food Application (CRUD API)

1. START

2. Import express module

3. Initialize app as an Express application

4. Configure app to parse JSON request bodies

 5. Start server

        a. Listen on port 2001

        b. Print "Food application server running on port 2001 "

 6. Define routes:

  a. GET "/"

        - Send "Welcome to the Food Application "
  b. GET "/menu"
          - Send "Here is the food menu with all available items"
 c. POST "/order"
          - Receive order details from request body
          - Send confirmation message with received order data
  d. PUT "/order/:id"
          - Receive order ID from URL parameter
          - Receive update data from request body
          - Send confirmation message with updated details
 e. DELETE "/order/:id"
          - Receive order ID from URL parameter
          - Send confirmation message that the order is cancelled
7. END
```

---

3. Pseudocode

```
START
 IMPORT express library
CREATE app using express
SET app to use JSON for request bodies
 START server on port 2001
DISPLAY "Food application server running on port 2001"
```

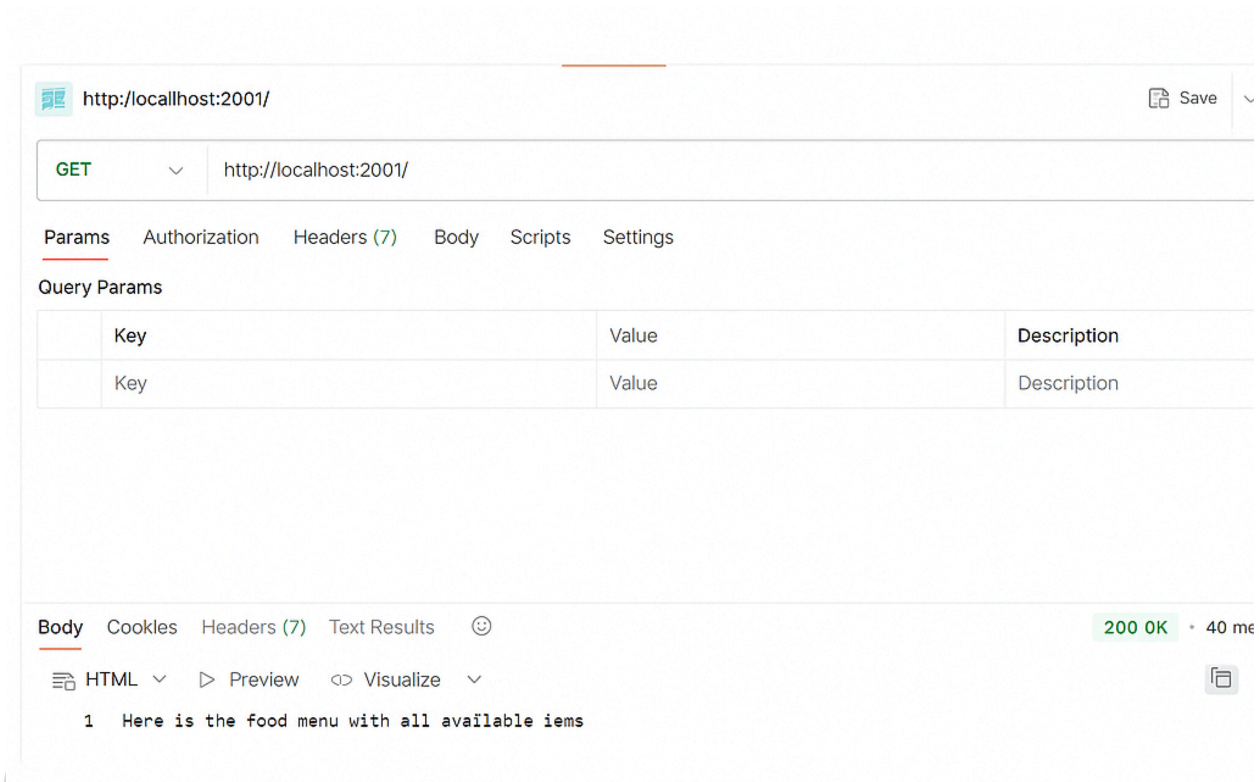
```
IF request is GET "/"
    SEND "Welcome to the Food Application"
IF request is GET "/menu"
    SEND "Here is the food menu with all available items"
IF request is POST "/order"
    READ order details from request body
    SEND "Order placed successfully for: " + order details
\IF request is PUT "/order/:id"
    READ order ID from request parameters
    READ updated details from request body
    SEND "Order with ID {id} updated with: " + updated details
IF request is DELETE "/order/:id"
    READ order ID from request parameters
    SEND "Order with ID {id} has been cancelled"
END
```

---

4. Program Code

```
var express = require("express");

var app = express();

 app.use(express.json());

 app.listen(2001, () => {

  console.log("Food application server running on port 2001");
});

 app.get("/", (req, res) => {

  res.send("Welcome to the Food Application ");

});

 app.get("/menu", (req, res) => {

  res.send("Here is the food menu with all available items");

});

 app.post("/order", (req, res) => {

  res.send(`Order placed successfully for: ${JSON.stringify(req.body)}`);

});

 app.put("/order/:id", (req, res) => {

  res.send(`Order with ID ${req.params.id} has been updated with:
${JSON.stringify(req.body)}`);

});

 app.delete("/order/:id", (req, res) => {

  res.send(`Order with ID ${req.params.id} has been cancelled`);

});
```

6. Screenshots of Output



http:/localhost:2001/

GET http://localhost:2001/

Params Authorization Headers (7) Body Scripts Settings

Query Params

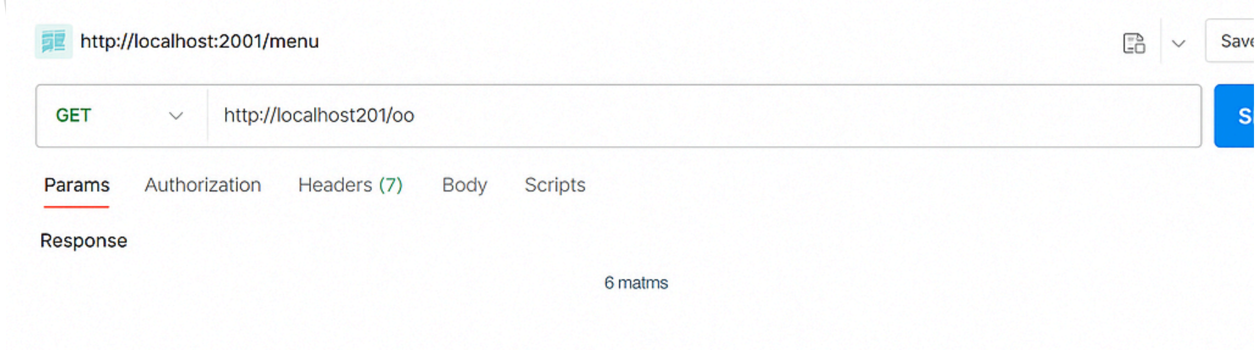
	Key	Value	Description
	Key	Value	Description

Body Cookies Headers (7) Text Results 200 OK • 40 me

HTML Preview Visualize

```
Here is the food menu with all avaïlable iems
```

Figure 2: Get output with /



http://localhost:2001/menu

GET http://localhost201/oo

Params Authorization Headers (7) Body Scripts

Response

6 matms

7. Observation / Reflection

This program is a simple Food Application backend built with Express.js. It demonstrates basic CRUD operations for handling food orders:

- **Create** (POST) → Place a new food order

- **Read** (GET) → View the menu and home page message
- **Update** (PUT) → Modify order details using an order ID
- **Delete** (DELETE) → Cancel an existing order