SESSION-2

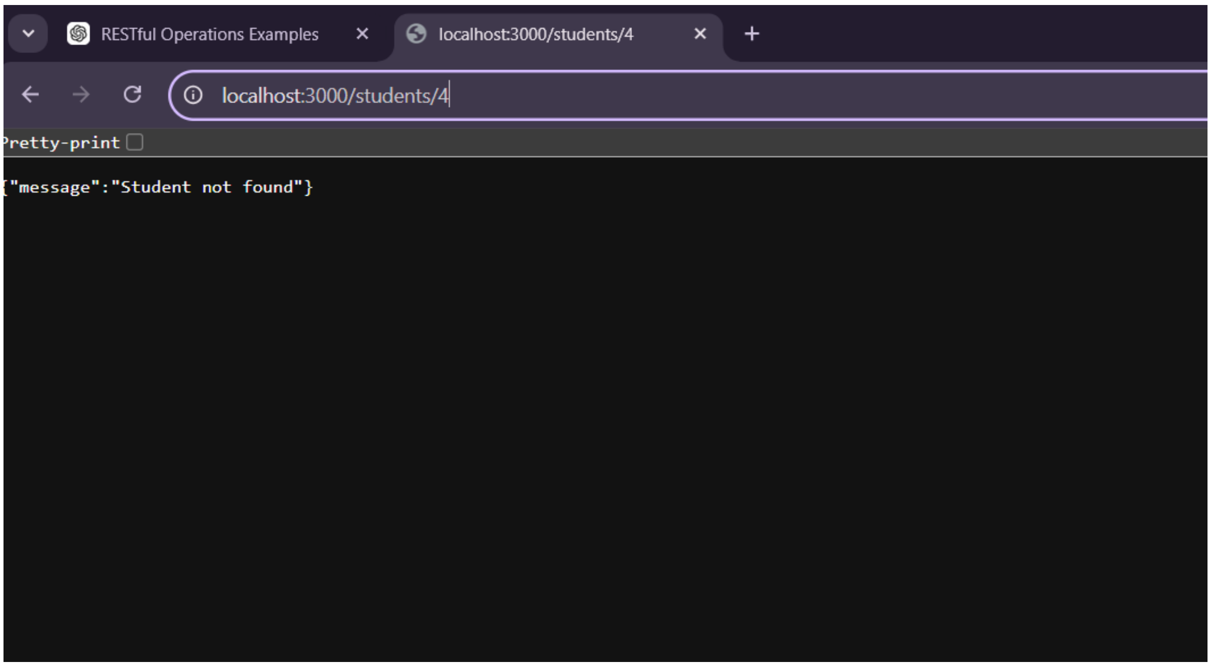
Name : V Rithik Reddy
H.no : 2303A51263
B-19

SCENARIO-1:

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

const app = express();


app.use(express.json());


let students = [

  { id: 1, name: "Ravi" },

  { id: 2, name: "Sita" },

  { id: 3, name: "John" }

];


// Welcome route

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

  res.json({ message: "Welcome to Student Information Server" });

});


// All students
```

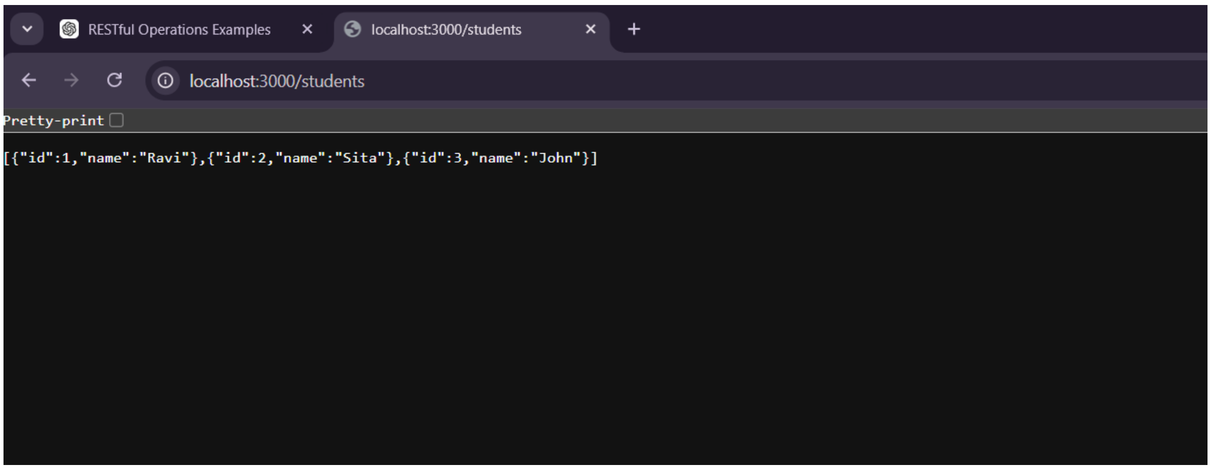
```
app.get('/students', (req, res) => {
  res.json(students);
});

// Student by ID
app.get('/students/:id', (req, res) => {
  const id = parseInt(req.params.id);
  const student = students.find(s => s.id === id);

  if (!student) {
    return res.status(404).json({ message: "Student not found" });
  }

  res.json(student);
});

app.listen(3000, () => {
  console.log("Server running on port 3000");
});
```

IF STUDENT NOT FOUND:


RESTful Operations Examples
localhost:3000/students/4
localhost:3000/students/4
Pretty-print

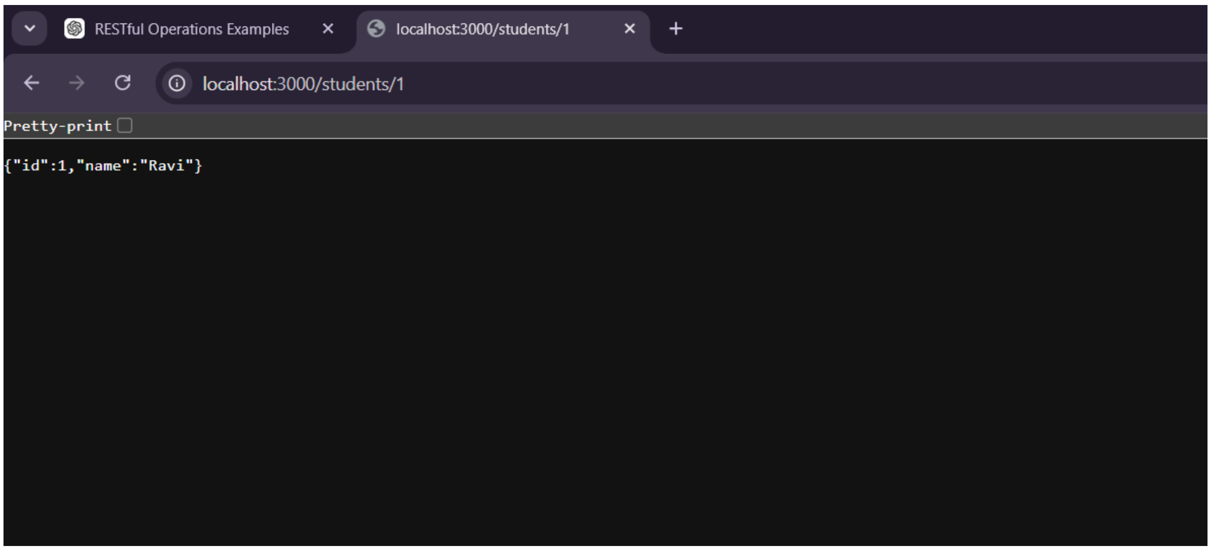
```
{"message":"Student not found"}
```

DISPLAY THE DETAILS OF STUDENTS:



```
[{"id":1,"name":"Ravi"},{"id":2,"name":"Sita"},{"id":3,"name":"John"}]
```

/students/:id → Displays details of a specific student:


RESTful Operations Examples
localhost:3000/students/1
localhost:3000/students/1
Pretty-print

```
{"id":1,"name":"Ravi"}
```

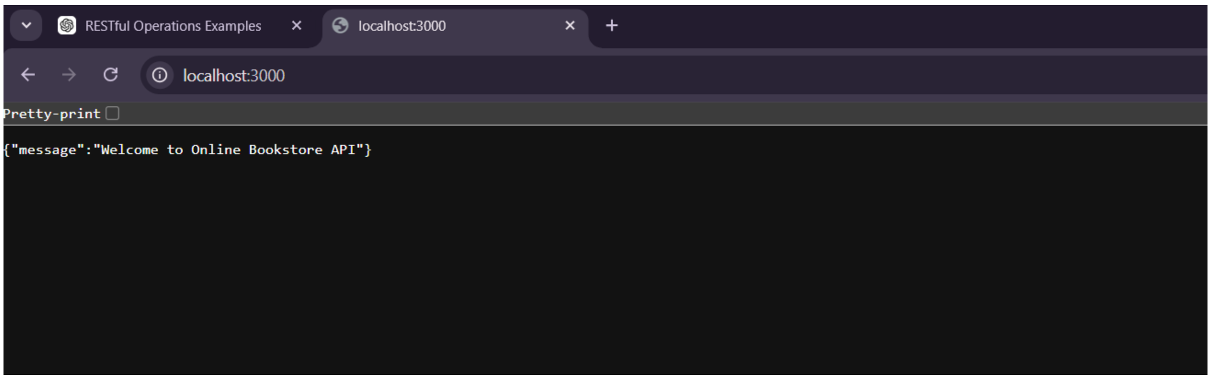
SCENARIO-2:

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

const app = express();


app.use(express.json());


let books = [];


// Welcome route

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

  res.json({ message: "Welcome to Online Bookstore API" });

});


// GET all books

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

  res.json(books);
```

```
});

// ADD new book
app.post('/books', (req, res) => {
  const { id, name } = req.body;

  if (!id || !name) {
    return res.status(400).json({ message: "ID and Name required" });
  }

  books.push({ id, name });

  res.status(201).json({
    message: "Book added successfully",
    books: books
  });
});

// UPDATE book
app.put('/books/:id', (req, res) => {
  const id = parseInt(req.params.id);

  const book = books.find(b => b.id === id);
  if (!book) {
    return res.status(404).json({ message: "Book not found" });
  }

  book.name = req.body.name;

  res.json({
    message: "Book updated successfully",
    book: book
  });
});

// DELETE book
app.delete('/books/:id', (req, res) => {
  const id = parseInt(req.params.id);

  books = books.filter(b => b.id !== id);

  res.json({ message: "Book deleted successfully" });
});

app.listen(3000, () => {
  console.log("Server running on port 3000");
```

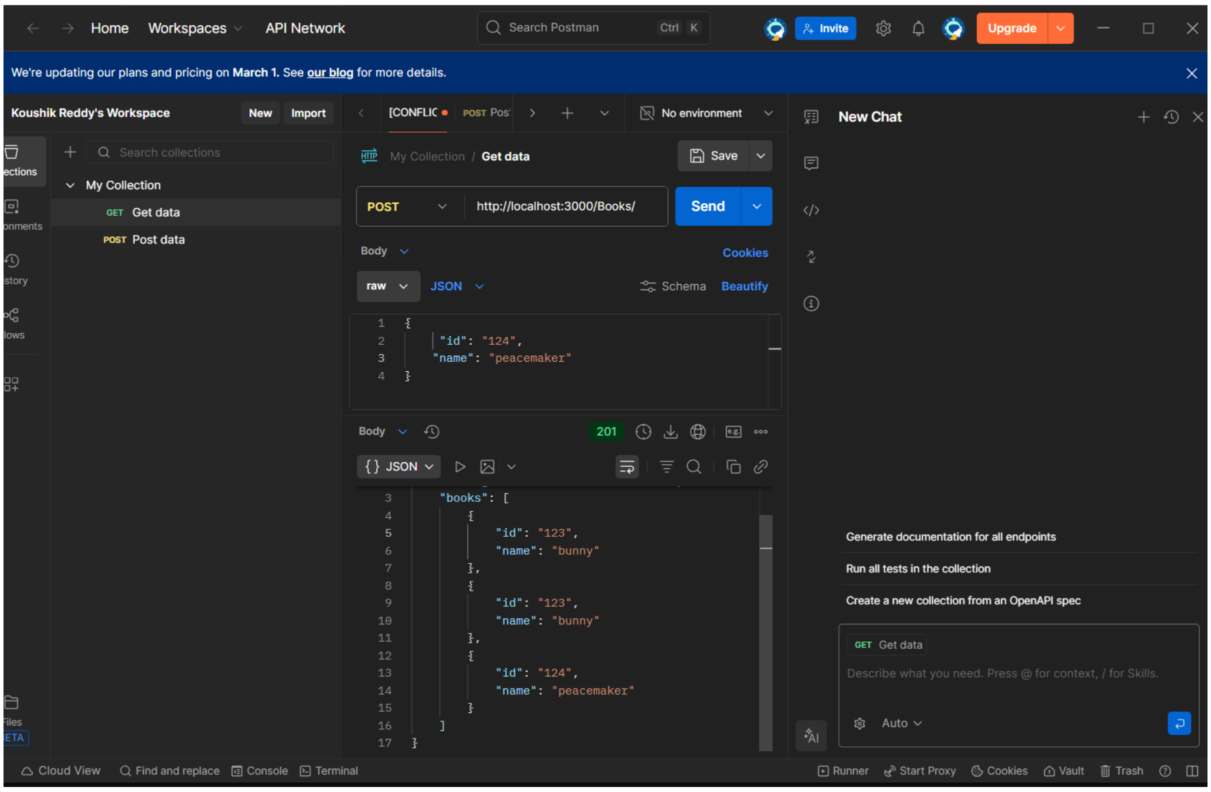
});

output:



```
{"message":"Welcome to Online Bookstore API"}
```

put operations:



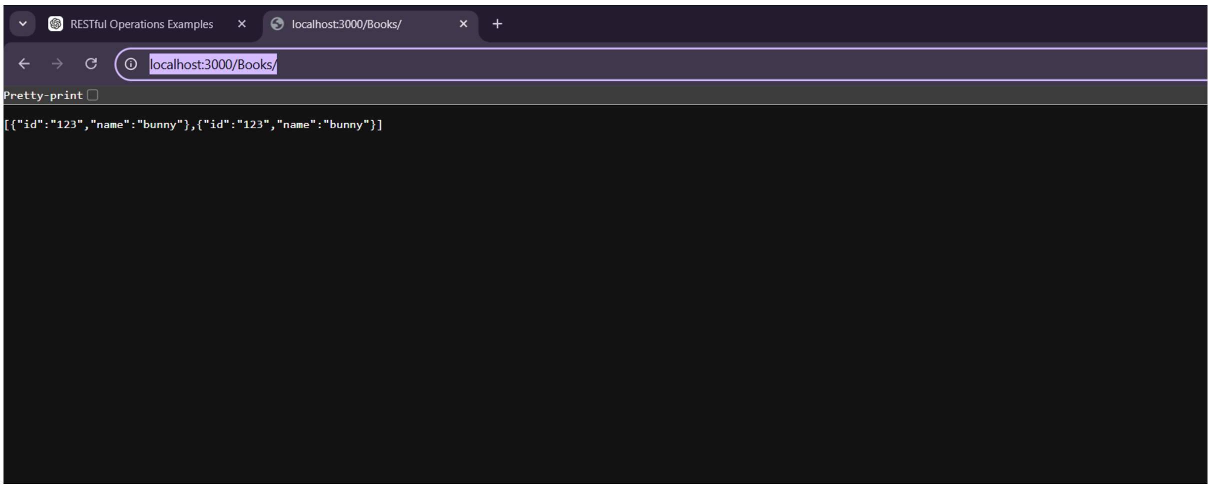
POST http://localhost:3000/Books/

```
{
    "id": "124",
   "name": "peacemaker"
}
```

201

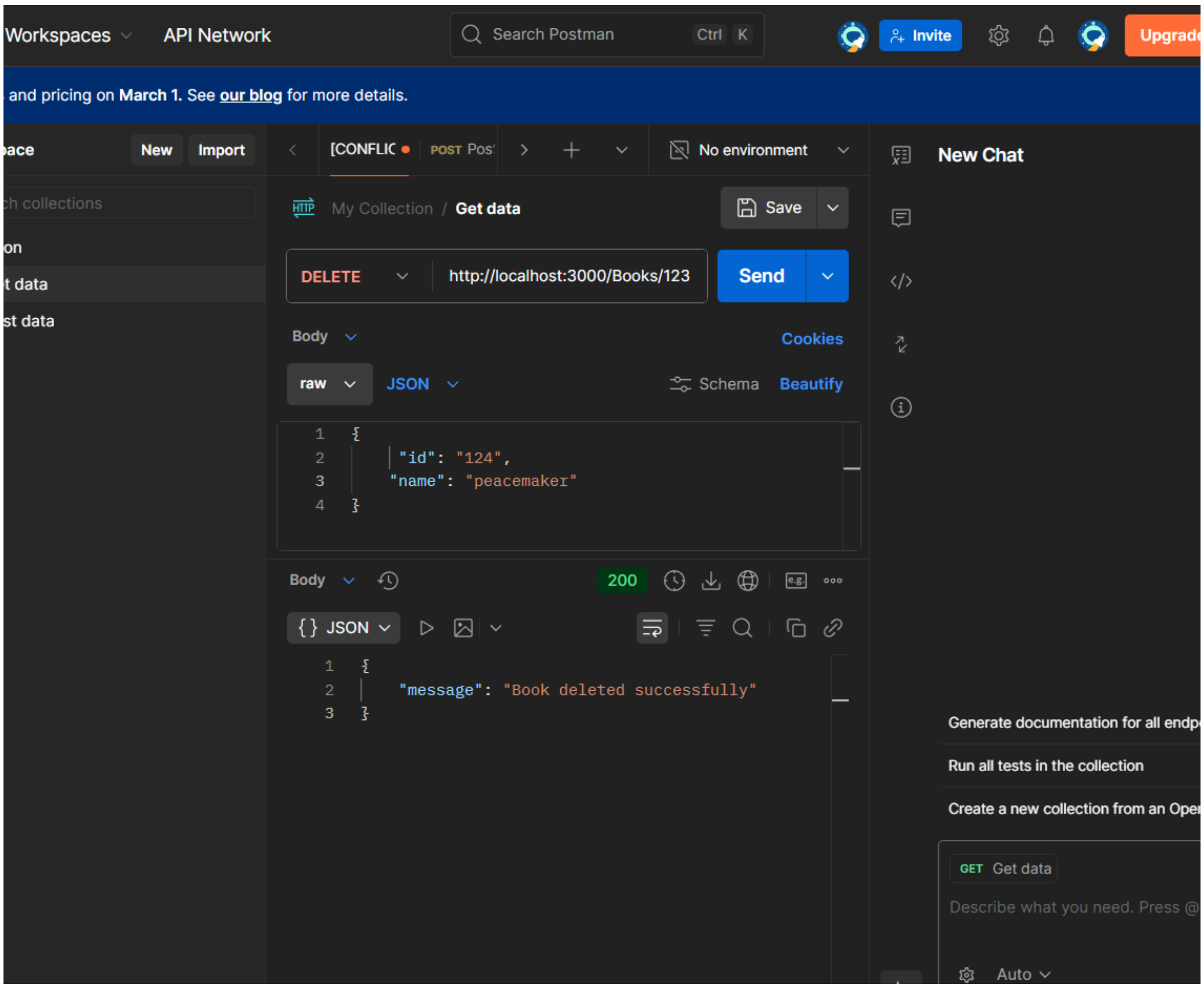
```
    "books": [
        {
            "id": "123",
            "name": "bunny"
        },
        {
            "id": "123",
            "name": "bunny"
        },
        {
            "id": "124",
            "name": "peacemaker"
        }
    ]
}
```

put output:



```
[{"id":"123","name":"bunny"},{"id":"123","name":"bunny"}]
```

Delete operations:

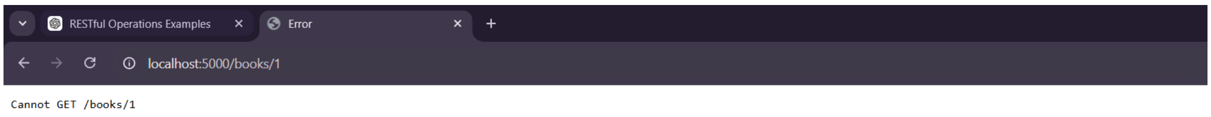

Workspaces
API Network
Search Postman
Invite
Upgrade
and pricing on March 1. See our blog for more details.
New
Import
No environment
New Chat
My Collection / Get data
Save
Send
Body
Cookies
raw
JSON
Schema
Beautify
200
JSON
Generate documentation for all endp
Run all tests in the collection
Create a new collection from an Ope
GET Get data
Describe what you need. Press @
Auto

DELETE http://localhost:3000/Books/123

```
{
    "id": "124",
    "name": "peacemaker"
}
```

```
{
    "message": "Book deleted successfully"
}
```

output:



```
Cannot GET /books/1
```

scenario -3:

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

const app = express();


app.use(express.json());


let users = [];


// ================================

// AUTHENTICATION MIDDLEWARE

// ================================

function authMiddleware(req, res, next) {

  const username = req.headers.username;


  if (!username) {

    return res.status(401).json({ message: "Access denied! Please
login." });

  }
  next();
}

// ================================
// REGISTER ROUTE
// ================================
app.post('/register', (req, res) => {
  const { username, password } = req.body;

  if (!username || !password) {
    return res.status(400).json({ message: "Username & Password
required" });
  }

  users.push({ username, password });

  res.json({
    message: "User registered successfully",
    users: users
  });
});

// ================================
// LOGIN ROUTE
// ================================
app.post('/login', (req, res) => {
  const { username, password } = req.body;

  const user = users.find(
    u => u.username === username && u.password === password
  );

  if (!user) {
    return res.status(401).json({ message: "Invalid login credentials" });
  }

  res.json({ message: "Login successful!" });
});

// ================================
// PROTECTED DASHBOARD ROUTE
// ================================
app.get('/dashboard', authMiddleware, (req, res) => {
  res.json({ message: "Welcome to Dashboard — You are logged in!"
});
});
```

```
app.listen(3000, () => {
  console.log("Server running on port 3000");
});
```