

RESTFUL OPERATIONS

Y.Sahasra
2303A51269
B-19

```
#### simple Restfull operation

const express = require('express');

const app = express();
app.use(express.json());

app.get('/students', (req, res) => {
  res.json([{ id: 1, name: "Ravi" }]);
});

app.post('/students', (req, res) => {
  const student = req.body;
  res.status(201).json({ message: "Student added", data: student });
});

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

#### restfull operation with HTML

const express = require('express');

const app = express();
app.use(express.urlencoded({ extended: true }));
app.use(express.json());

let students = [];

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

<h2>Add Student</h2>

<form method="POST" action="/students">
`)
```

```
<input type="text" name="name" placeholder="Enter name" />
<button type="submit">Add</button>
</form>
`);

});

app.post('/students', (req, res) => {
const name = req.body.name;
const newStudent = {
id: students.length + 1,
name: name
};
students.push(newStudent);
res.send("Student Added Successfully!");
});

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

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

#### resfull operation with POSTMAN desktop app(https://www.postman.com/downloads/)
const express = require('express');
const app = express();
app.use(express.json());
let students = [
{ id: 1, name: "Ravi" },
{ id: 2, name: "Sita" }
];
// 1 GET - View all students
```

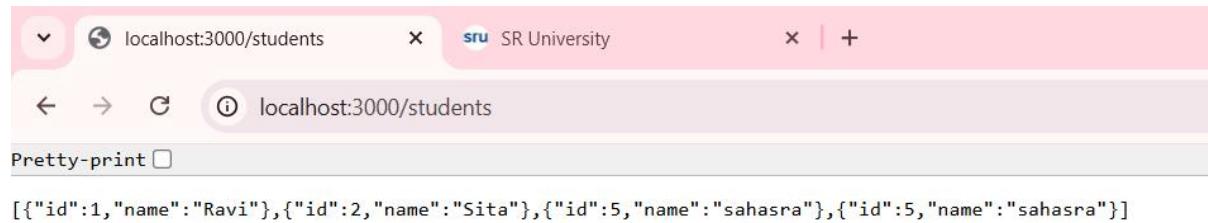
```
app.get('/students', (req, res) => {  
  res.json(students);  
});
```

GET OPERATION:

/ 1 GET - View all students

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

OUTPUT:



A screenshot of the Postman application interface. The left sidebar shows 'YADA SAHASRA's Workspace' with 'Collections', 'Environments', 'History', 'Flows', and 'Files (BETA)'. The main area shows a 'My Collection / Post data' section. A 'POST Post data' tab is selected, with the URL 'http://localhost:3000/students/6'. The 'Body' tab is active, showing a raw JSON payload: {"name": "sahasra"}. The response at the bottom shows a 200 OK status with the message: {"message": "Student deleted"}

```
// 2 GET - View single student

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);
});
```

OUTPUT:

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'Collections' (selected), 'Environments', 'History', 'Flows', and 'Files (BETA)'. The main area shows a 'My Collection / Get data' section with a 'GET' request to 'http://localhost:3000/students'. The 'Params' tab is selected, showing a table for 'Query Params' with one row: 'Key' and 'Value'. Below the request, the 'Body' tab is selected, showing a JSON table with four rows of student data:

	id	name
0	1	Ravi
1	2	Sita
2	5	sahasra
3	5	sahasra

/ 4 PUT - Update student

```
app.put('/students/:id', (req, res) => {
```

```
const id = parseInt(req.params.id);

const { name } = req.body;

const student = students.find(s => s.id === id);

if (!student) {

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

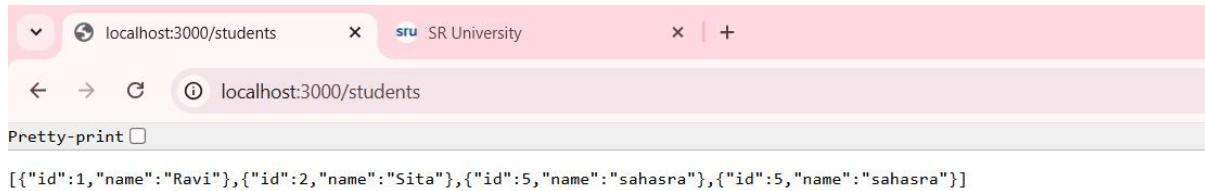
}

student.name = name;

res.json({ message: "Student updated", student });

});
```

OUTPUT:



The screenshot shows the Postman application interface. On the left, there's a sidebar with 'YADA SAHASRA's Workspace' containing sections for Collections, Environments, History, Flows, and Files (BETA). The main area shows a collection named 'My Collection' with a 'Post data' item selected. A PUT request is being made to 'http://localhost:3000/students'. The request body is set to raw JSON with the value: { "name": "sahasra" }. The response status is '404 Not Found' with a timestamp of '11 ms' and a size of '419 B'. The response body is an HTML error page with the following content:

```

1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="utf-8">
6      <title>Error</title>
7  </head>
8
9  <body>
10     <p>Cannot PUT /students</p>
11 </body>

```

// 5 DELETE - Remove student

```

app.delete('/students/:id', (req, res) => {
  const id = parseInt(req.params.id);
  students = students.filter(s => s.id !== id);
  res.json({ message: "Student deleted" });
});

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

```

OUTPUT:

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'YADA SAHASRA's Workspace' containing sections for Collections, Environments, History, Flows, and Files (BETA). The main workspace is titled 'My Collection / Post data'. A 'DELETE' request is being made to the URL `http://localhost:3000/students/6`. The 'Body' tab is selected, showing the JSON payload:

```
1 {  
2   "name": "sahasra"  
3 }
```

Below the request, the response details are shown: **200 OK**, 7 ms, 264 B. The response body is also a JSON object:

```
1 {  
2   "message": "Student deleted"  
3 }
```

The screenshot shows a browser window with the address bar set to `localhost:3000/students`. The page content is a JSON array of student objects:

```
[{"id": 1, "name": "Ravi"}, {"id": 2, "name": "Sita"}, {"id": 5, "name": "sahasra"}, {"id": 5, "name": "sahasra"}]
```