DAY-9

Introduction to Backend Development and Basics of Node.js

- ***What is Backend Development?***

  - Backend development is like the behind-the-scenes work of a website or app. It includes everything that happens on the server, such as:
    1. **Managing Data:** The backend stores and retrieves information. For example, when you log into a website, the backend checks your username and password from a database.
    2. **Server Logic:** It handles the requests you make from the front end (the part you see on the screen) and sends back the right information.
    3. **APIs:** These are like messengers between the front end and backend, helping the two communicate with each other.

<Think of the frontend as the tip of an iceberg (what users see) and the backend as everything under the water that makes it work.>

- ***What is Node.js?***
  - Node.js is a tool that allows developers to run JavaScript (the programming language used for websites) on the backend (server side). Normally, JavaScript runs in your browser (for example, when you're interacting with a website), but Node.js lets you use JavaScript to build the server-side part of a website or app too.
  - So, instead of using different languages for frontend and backend, with Node.js, you can use JavaScript for both, which simplifies things.
  - ***Key Things About Node.js:***
    1. **Fast and Scalable:** It's really fast because it can handle many requests at the same time without getting stuck waiting for one to finish.
    2. **Non-Blocking:** This means if one task (like accessing a database) takes time, Node.js doesn't stop other tasks. It just keeps going.
    3. **Uses JavaScript:** If you know JavaScript for building websites, you can also use it to build the server part, which makes it easier to learn and work with.

DAY-9

BOOK-COLLECTION

server.js file : (java script file)

Code:

```
const express = require('express');
const app = express();
const port = 3000;

app.use(express.json()); // Middleware to parse JSON bodies

// In-memory "database"
let books = [];
let nextId = 1;

// Routes

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

// Get a book by ID
app.get('/books/:id', (req, res) => {
  const bookId = parseInt(req.params.id);
  const book = books.find(b => b.id === bookId);
  if (book) {
    res.json(book);
  } else {
    res.status(404).json({ message: 'Book not found' });
  }
});

// Add a new book
app.post('/books', (req, res) => {
  const { title, author, year } = req.body;
  if (!title || !author || !year) {
    return res.status(400).json({ message: 'Title, author, and year are required' });
  }
  const newBook = { id: nextId++, title, author, year };
  books.push(newBook);
```

DAY-9

```
  res.status(201).json(newBook);
});

// Update a book by ID
app.put('/books/:id', (req, res) => {
  const bookId = parseInt(req.params.id);
  const bookIndex = books.findIndex(b => b.id === bookId);
  if (bookIndex === -1) {
    return res.status(404).json({ message: 'Book not found' });
  }

  const { title, author, year } = req.body;
  books[bookIndex] = { id: bookId, title: title || books[bookIndex].title, author: author ||
books[bookIndex].author, year: year || books[bookIndex].year };
  res.json(books[bookIndex]);
});

// Delete a book by ID
app.delete('/books/:id', (req, res) => {
  const bookId = parseInt(req.params.id);
  const bookIndex = books.findIndex(b => b.id === bookId);
  if (bookIndex === -1) {
    return res.status(404).json({ message: 'Book not found' });
  }

  books.splice(bookIndex, 1);
  res.status(204).send(); // No content
});

// Start the server
app.listen(port, () => {
  console.log(`Server running at http://localhost:${port}`);
});
```

- ***Run the Server:***

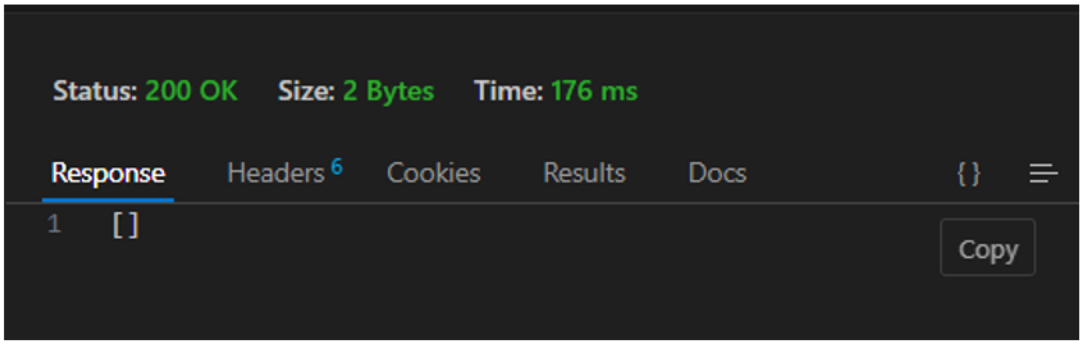
```
node server.js
Server running at http://localhost:1000
```

DAY-9

Test the API Endpoints

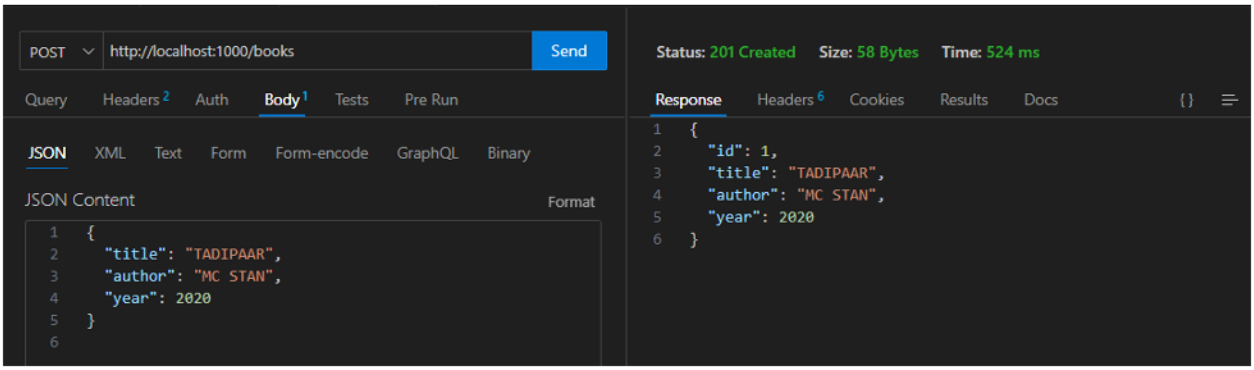
1. ***GET /books: Retrieve all books***

- ***Output:***



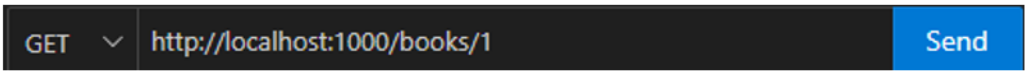
2. ***POST /books: Add a new book***

- ***Output:***

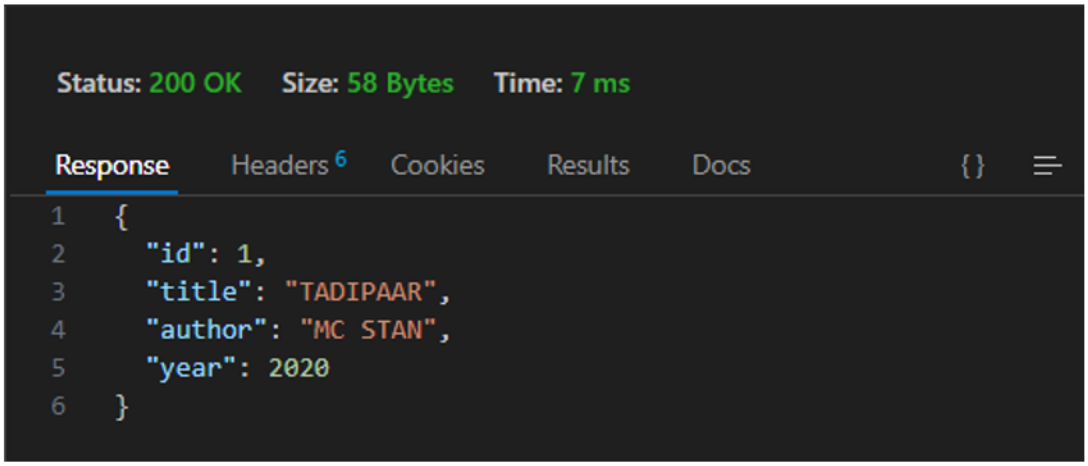


DAY-9

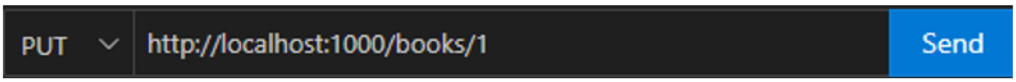
3. *GET /books/:id: Retrieve a specific book by ID*



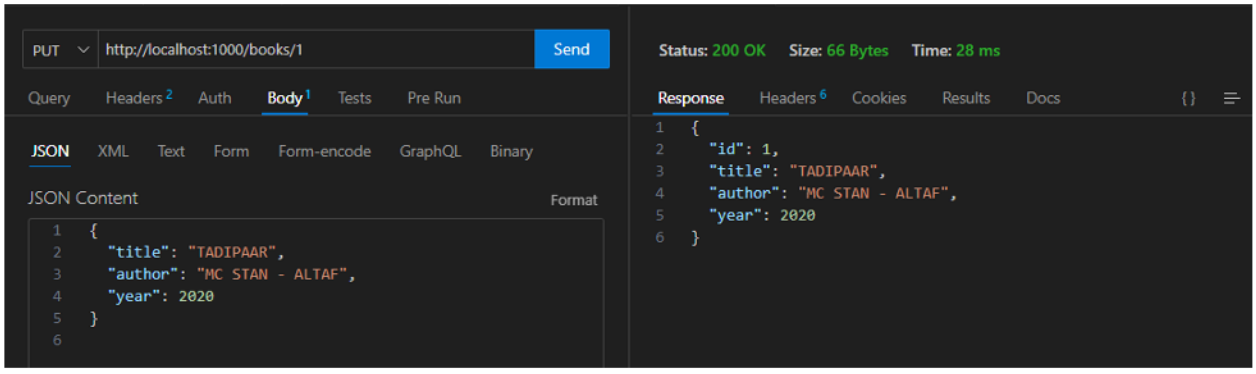
- ***Output:***



4. *PUT /books/:id: Update a book by ID*

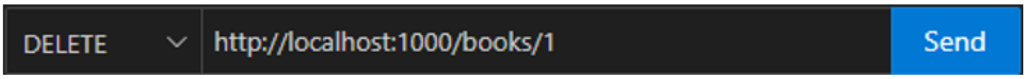


- ***Output:***

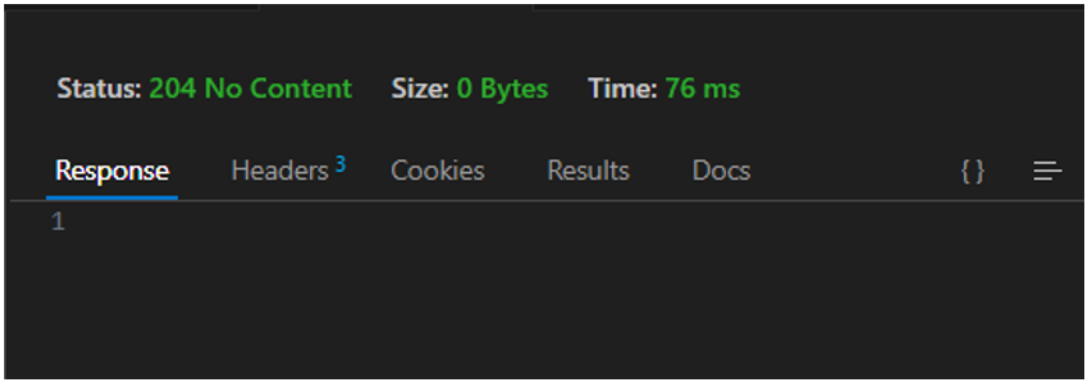


DAY-9

5. ***DELETE /books/:id: Delete a book by ID***



- ***Output:***



6. ***Verify Deletion:***
   - *GET /books again:*

   - ***Output:***