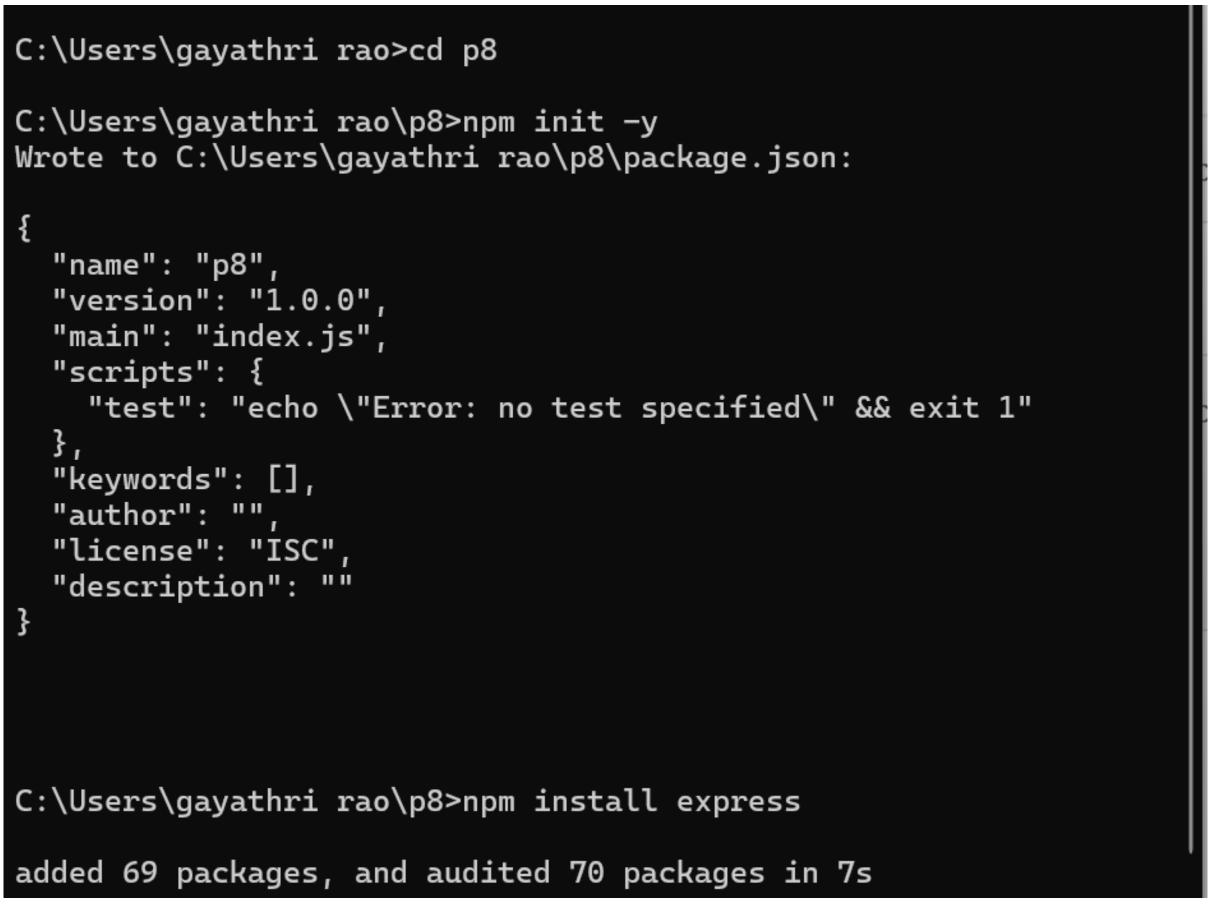
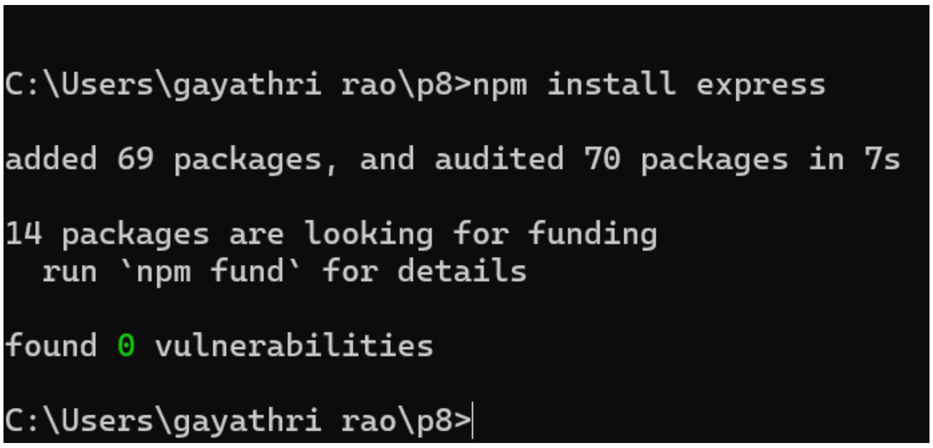
8. Build a REST API to Fetch Specific Article (GET) and updating articles using PUT and PATCH methods using Express a. Implement the route /articles/:id to fetch article by ID. b. Implement a PUT route /article/:id to update the entire article. c. Implement a PATCH route /article/:id to update partial fields of the article. d. Test both endpoints via Postman by updating existing articles.

Index.js

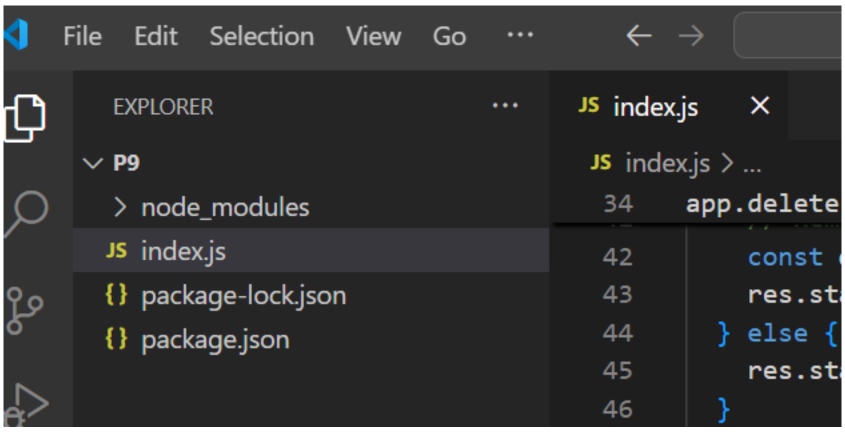
```
const express = require('express');

const app = express();

const port = 3000;


// Middleware to parse JSON request bodies

app.use(express.json());


// Simulating a simple in-memory database of articles

let articles = [

  { id: 1, title: 'Article 1', content: 'This is the first article' },

  { id: 2, title: 'Article 2', content: 'This is the second article' },

  { id: 3, title: 'Article 3', content: 'This is the third article' },

];


// GET: Fetch article by ID

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

  const articleId = parseInt(req.params.id, 10);

  const article = articles.find(a => a.id === articleId);


  if (article) {

    res.status(200).json(article);

  } else {

    res.status(404).json({ message: 'Article not found' });
  }
});

// PUT: Update the entire article
app.put('/article/:id', (req, res) => {
  const articleId = parseInt(req.params.id, 10);
  const { title, content } = req.body;

  const articleIndex = articles.findIndex(a => a.id === articleId);

  if (articleIndex !== -1) {
    articles[articleIndex] = { id: articleId, title, content };
    res.status(200).json(articles[articleIndex]);
  } else {
    res.status(404).json({ message: 'Article not found' });
  }
});

// PATCH: Update partial fields of the article
app.patch('/article/:id', (req, res) => {
  const articleId = parseInt(req.params.id, 10);
  const { title, content } = req.body;

  const article = articles.find(a => a.id === articleId);

  if (article) {
    if (title) article.title = title;
    if (content) article.content = content;
    res.status(200).json(article);
  } else {
    res.status(404).json({ message: 'Article not found' });
  }
});

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



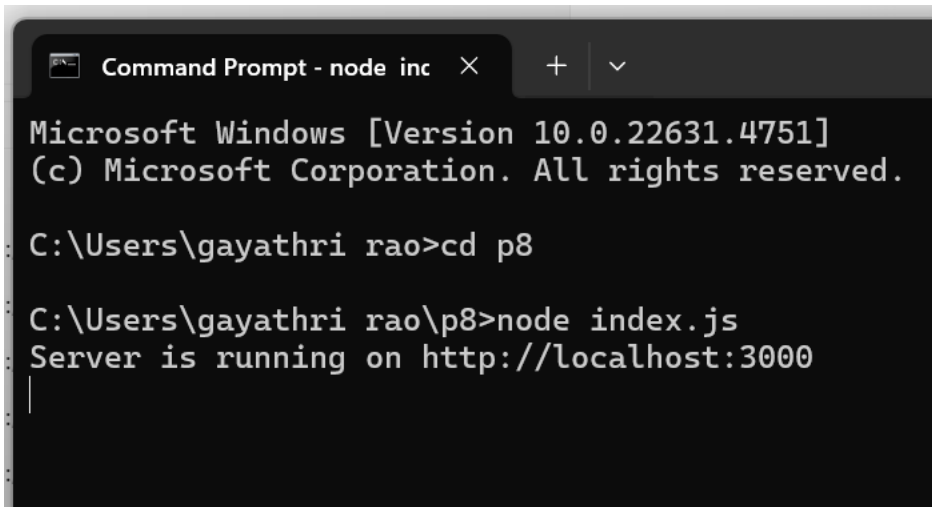
```
C:\Users\gayathri rao>cd p8

C:\Users\gayathri rao\p8>npm init -y
Wrote to C:\Users\gayathri rao\p8\package.json:

{
  "name": "p8",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}



C:\Users\gayathri rao\p8>npm install express

added 69 packages, and audited 70 packages in 7s
```



```
C:\Users\gayathri rao\p8>npm install express

added 69 packages, and audited 70 packages in 7s

14 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Users\gayathri rao\p8>
```

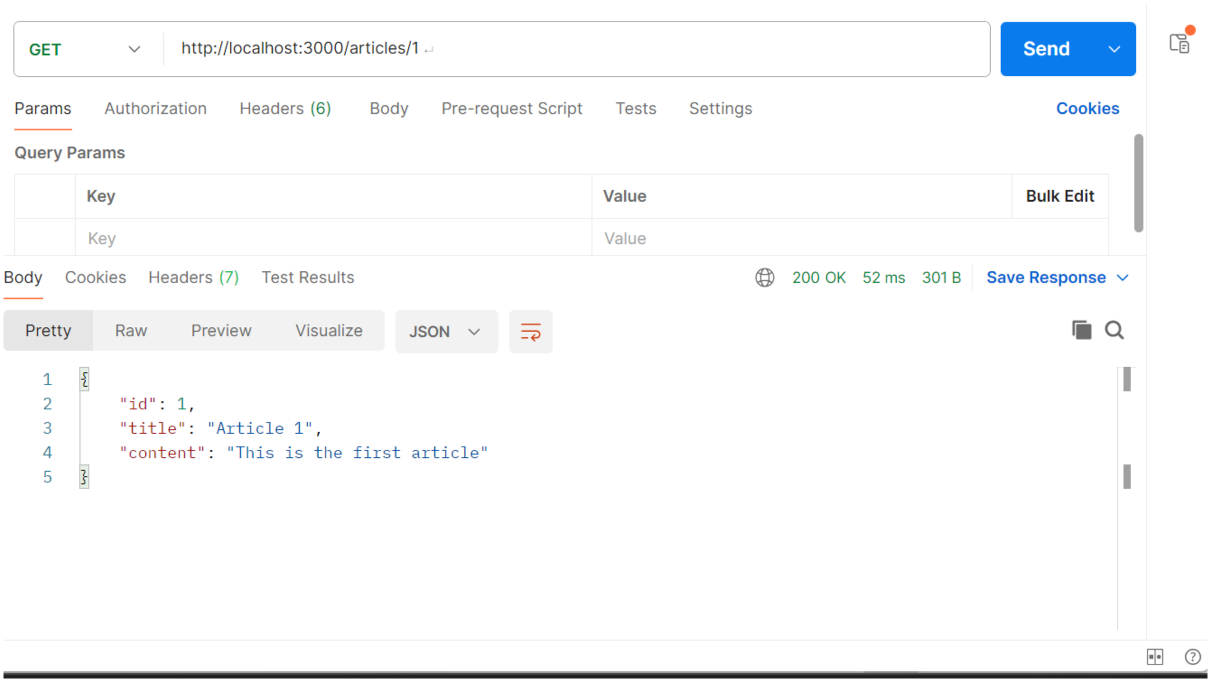


```
app.delete
    const
    res.st
  } else {
    res.st
  }
```



```
Microsoft Windows [Version 10.0.22631.4751]
(c) Microsoft Corporation. All rights reserved.

C:\Users\gayathri rao>cd p8

C:\Users\gayathri rao\p8>node index.js
Server is running on http://localhost:3000
```

GET:



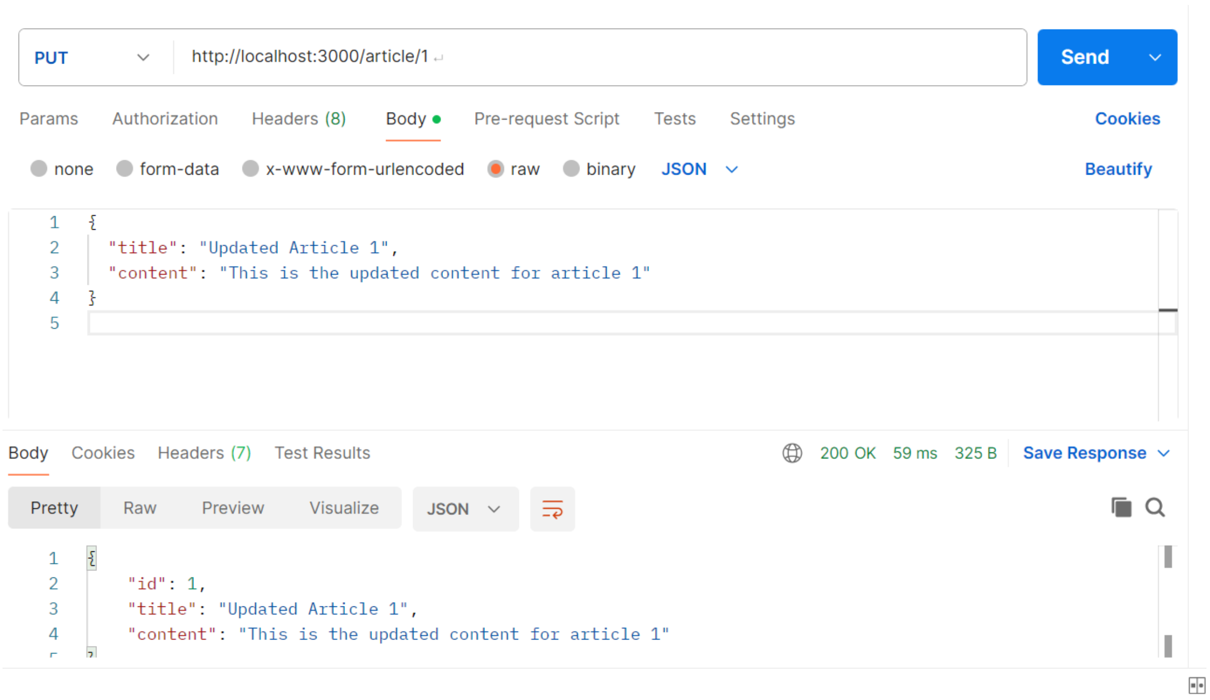
http://localhost:3000/articles/1

Key	Value
Key	Value

200 OK 52 ms 301 B

```
{
    "id": 1,
    "title": "Article 1",
    "content": "This is the first article"
}
```

PUT:

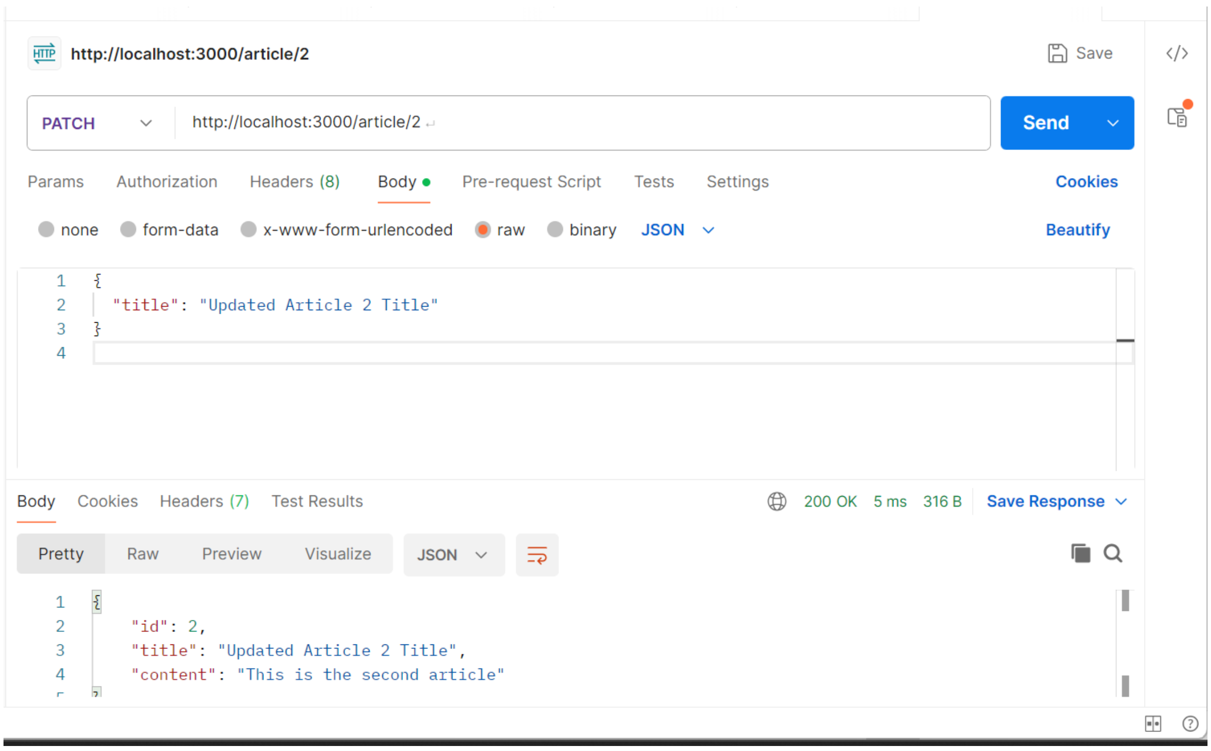


http://localhost:3000/article/1

```
{
  "title": "Updated Article 1",
  "content": "This is the updated content for article 1"
}
```

```
{
    "id": 1,
    "title": "Updated Article 1",
    "content": "This is the updated content for article 1"
```

PATCH:



http://localhost:3000/article/2

http://localhost:3000/article/2

```
{
  "title": "Updated Article 2 Title"
}
```

```
{
    "id": 2,
    "title": "Updated Article 2 Title",
    "content": "This is the second article"
```

9. Build a REST API to Creating new Article, Delete a Single Article (Row) and all the Articles (Rows) with POST and DELETE methods using Express a. Create a DELETE route /article/:id to delete single article by id. b. Test deletion of an article and handle cases when the article

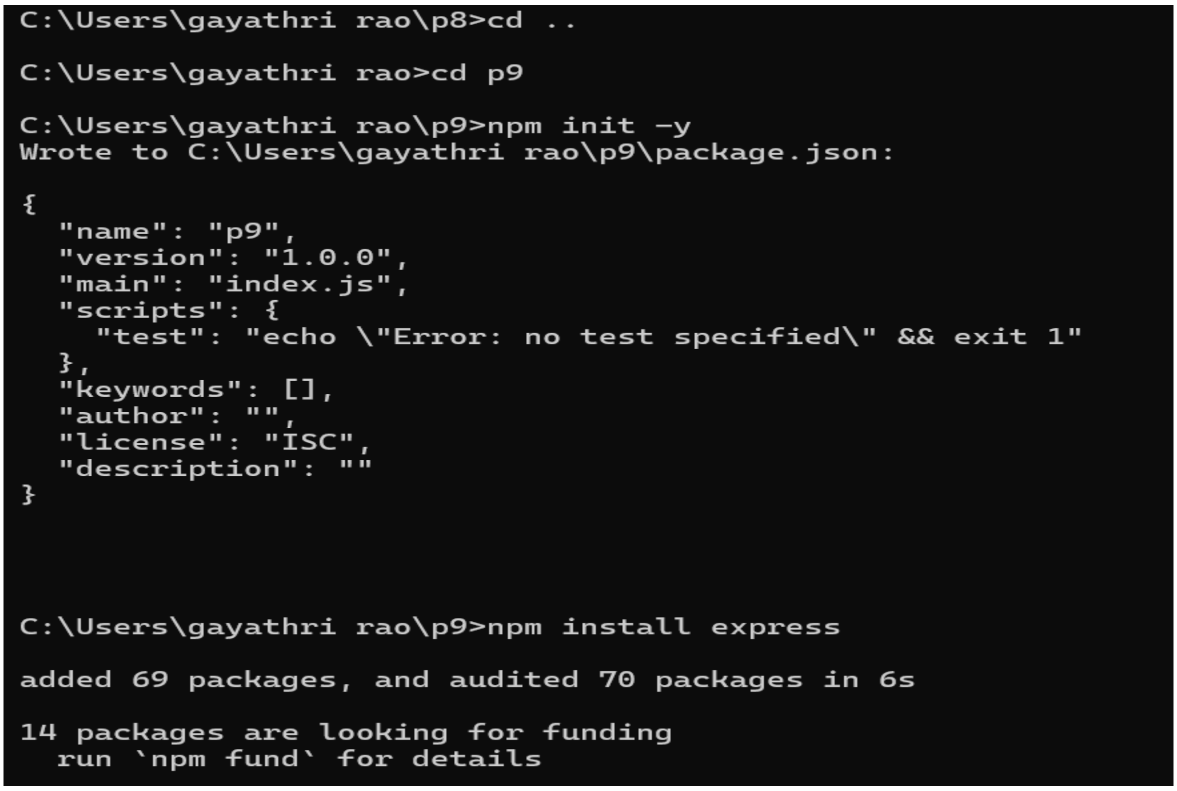
does not exist. c. Create DELETE route /articles to delete all articles. d. Test the API via Postman and verify deletion by calling GET /articles (should return empty array).

Index.js

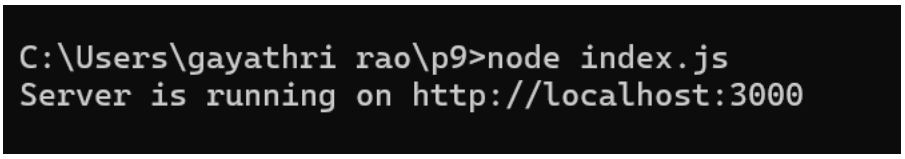
```
  const express = require('express');
const app = express();
const port = 3000;

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

// Simulating a simple in-memory database of articles
let articles = [
 { id: 1, title: 'Article 1', content: 'This is the first article' },
 { id: 2, title: 'Article 2', content: 'This is the second article' },
 { id: 3, title: 'Article 3', content: 'This is the third article' },
];

// POST: Create a new article
app.post('/article', (req, res) => {
 const { title, content } = req.body;

 // Generate a new ID for the article
 const newArticle = {
  id: articles.length + 1, // simple logic to generate ID
  title,
  content
 };
```

```
  // Add the new article to the articles array
  articles.push(newArticle);

  // Respond with the created article
  res.status(201).json(newArticle);
});

// DELETE: Delete a single article by ID
app.delete('/article/:id', (req, res) => {
  const articleId = parseInt(req.params.id, 10);

  // Find the index of the article to delete
  const articleIndex = articles.findIndex(a => a.id === articleId);

  if (articleIndex !== -1) {
    // Remove the article from the array
    const deletedArticle = articles.splice(articleIndex, 1);
    res.status(200).json(deletedArticle);
  } else {
    res.status(404).json({ message: 'Article not found' });
  }
});

// DELETE: Delete all articles
app.delete('/articles', (req, res) => {
  articles = []; // Clear all articles from the array
  res.status(200).json({ message: 'All articles deleted' });
});
```

```
// GET: Fetch all articles (for testing purposes)

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

  res.status(200).json(articles);

});


// Start the server

app.listen(port, () => {

  console.log(`Server is running on http://localhost:${port}`);

});
```

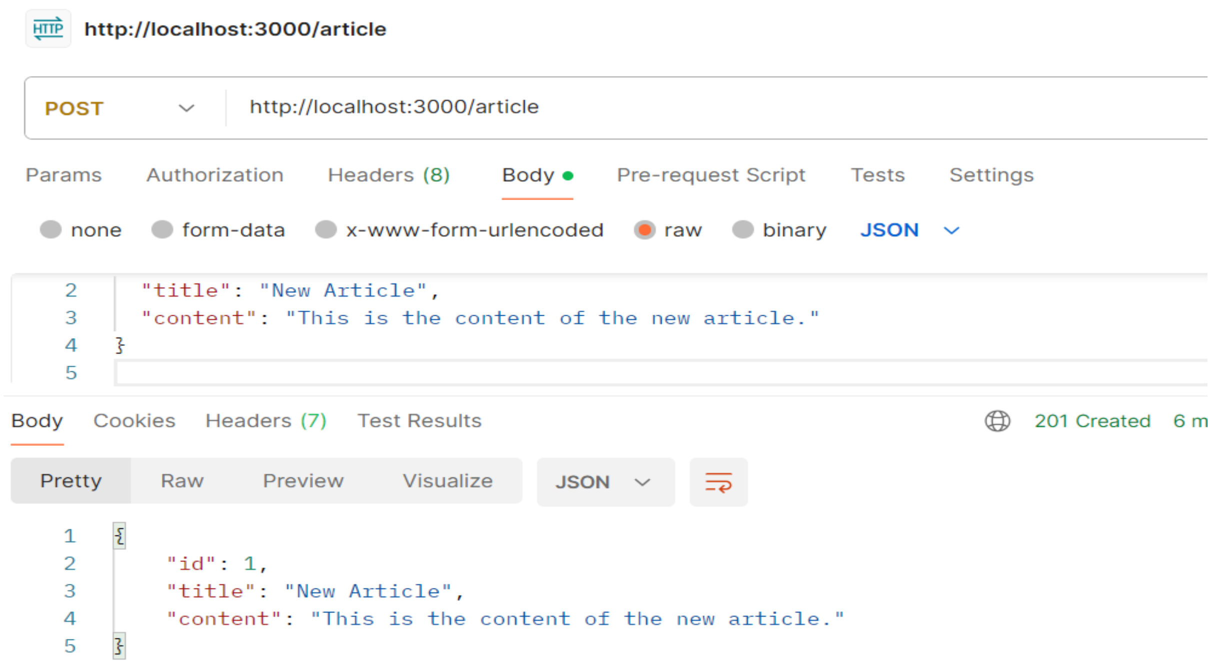


```
C:\Users\gayathri rao\p8>cd ..

C:\Users\gayathri rao>cd p9

C:\Users\gayathri rao\p9>npm init -y
Wrote to C:\Users\gayathri rao\p9\package.json:

{
  "name": "p9",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}



C:\Users\gayathri rao\p9>npm install express

added 69 packages, and audited 70 packages in 6s

14 packages are looking for funding
  run `npm fund` for details
```



```
C:\Users\gayathri rao\p9>node index.js
Server is running on http://localhost:3000
```

POST:


http://localhost:3000/article

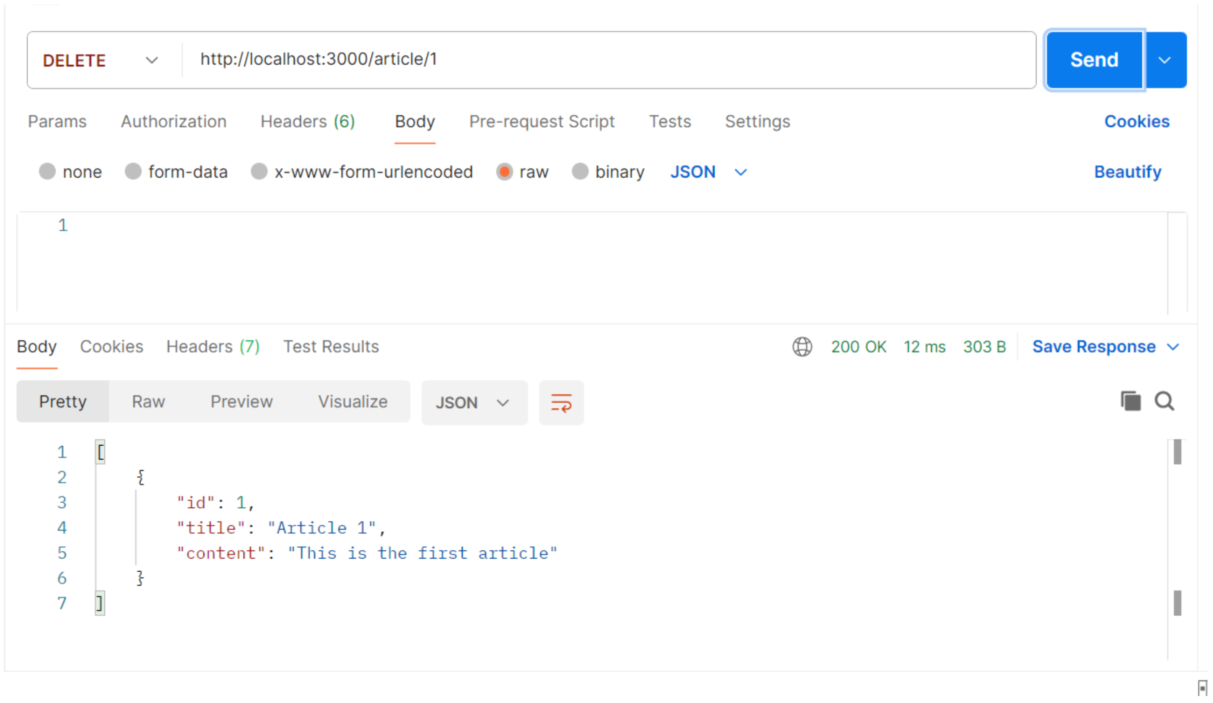
POST http://localhost:3000/article

```
  "title": "New Article",
  "content": "This is the content of the new article."
}
```

201 Created 6 m

```
{
    "id": 1,
    "title": "New Article",
    "content": "This is the content of the new article."
}
```

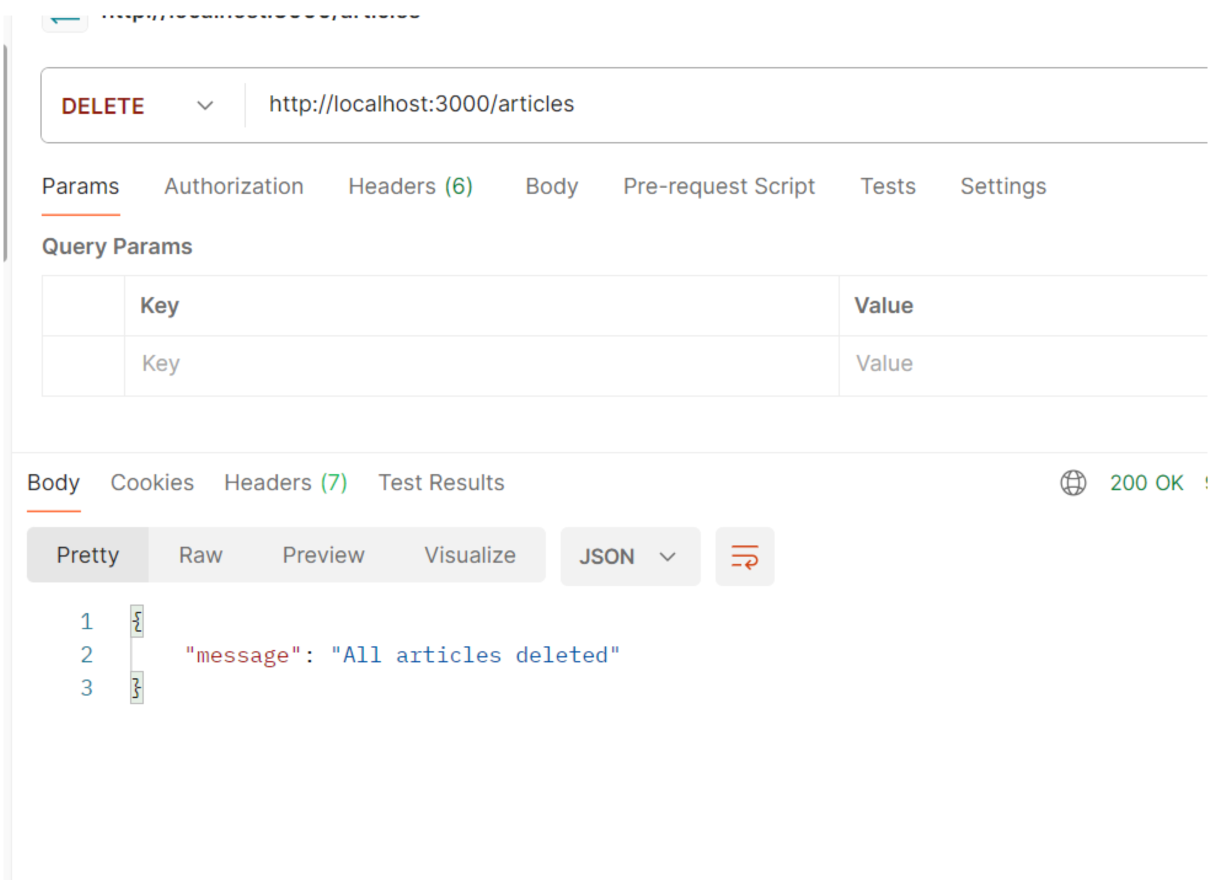
DELETE:


DELETE http://localhost:3000/article/1

200 OK 12 ms 303 B

```
[
    {
        "id": 1,
        "title": "Article 1",
        "content": "This is the first article"
    }
]
```

DELETE ALL:


DELETE http://localhost:3000/articles

Params | Authorization | Headers (6) | Body | Pre-request Script | Tests | Settings

Query Params

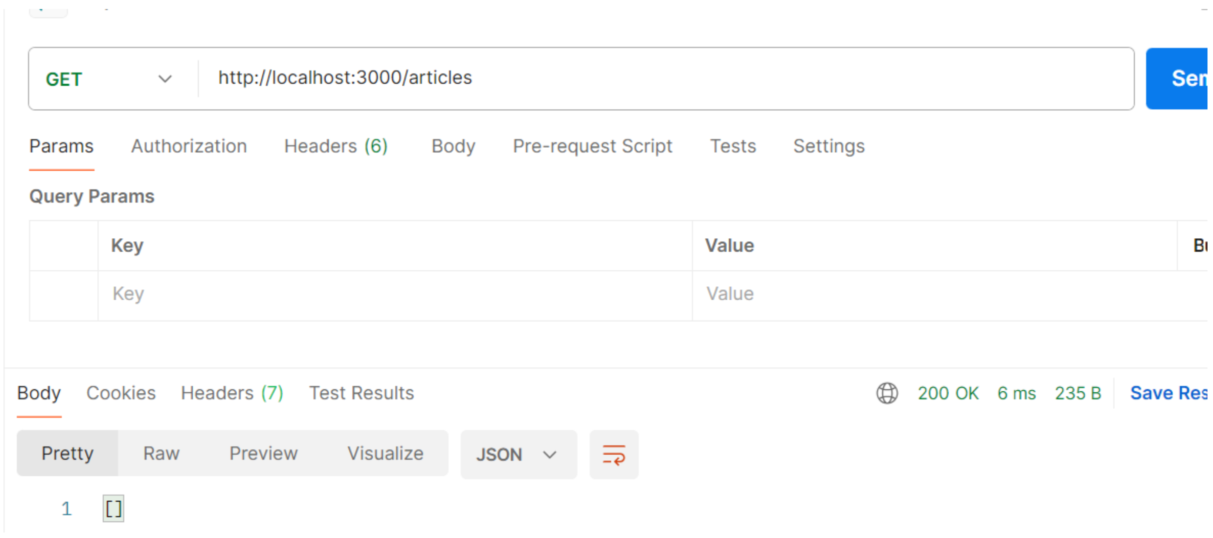
	Key	Value
	Key	Value

Body | Cookies | Headers (7) | Test Results — 200 OK

Pretty | Raw | Preview | Visualize | JSON

```
{
    "message": "All articles deleted"
}
```

GET: Fetch All Articles (Verify Deletion):


GET http://localhost:3000/articles

Params | Authorization | Headers (6) | Body | Pre-request Script | Tests | Settings

Query Params

	Key	Value
	Key	Value

Body | Cookies | Headers (7) | Test Results — 200 OK 6 ms 235 B

Pretty | Raw | Preview | Visualize | JSON

```
[]
```

POST /article: Create a new article.

DELETE /article/:id: Delete a single article by ID.

DELETE /articles: Delete all articles.

GET /articles: Fetch all articles.