EXP:4.2[11]

Deepak Saini [23BIS70072]

**Title**

REST API for Playing Card Collection Using Express.js

**Objective**

Build a RESTful API using Express.js to manage a collection of playing cards. This task helps you understand routing, handling HTTP methods, and basic data manipulation in a Node.js backend environment.

**Task Description**

Create an Express.js server that provides API endpoints to manage a playing card collection. The API should allow you to perform operations such as listing all cards, adding a new card (with properties like suit and value), retrieving a specific card by ID, and deleting a card by ID. Store card data in an in-memory array for simplicity. The API should follow RESTful principles and handle different HTTP methods (GET, POST, DELETE) clearly, responding with appropriate JSON data.

PROCEDURE:

```
const express=require("express");
const app = express();
app.use(express.json());

let cards = [
{ id: 1, suit: 'hearts', value: 'ace', collection: 'standard' },
{ id: 2, suit: 'spades', value: 'king', collection: 'vintage' }
];
```

```
function nextId(){
   let x=cards.length+1;
   return x;
}
app.get('/api/cards', (req, res) => {
   res.json(cards);
});

app.get('/api/cards/:id', (req, res) => {
   const card = cards.find(c => c.id === parseInt(req.params.id));
   if (!card) return res.status(404).json({ error: 'Card not found' });
   res.json(card);
});

app.post('/api/cards', (req, res) => {
   const { suit, value, collection } = req.body;
   if (!suit || !value || !collection) {
      return res.status(400).json({ error: 'Missing fields' });
   }
   const newCard = {
   id: nextId(),
   suit,
   value,
   collection
   };
   cards.push(newCard);
```

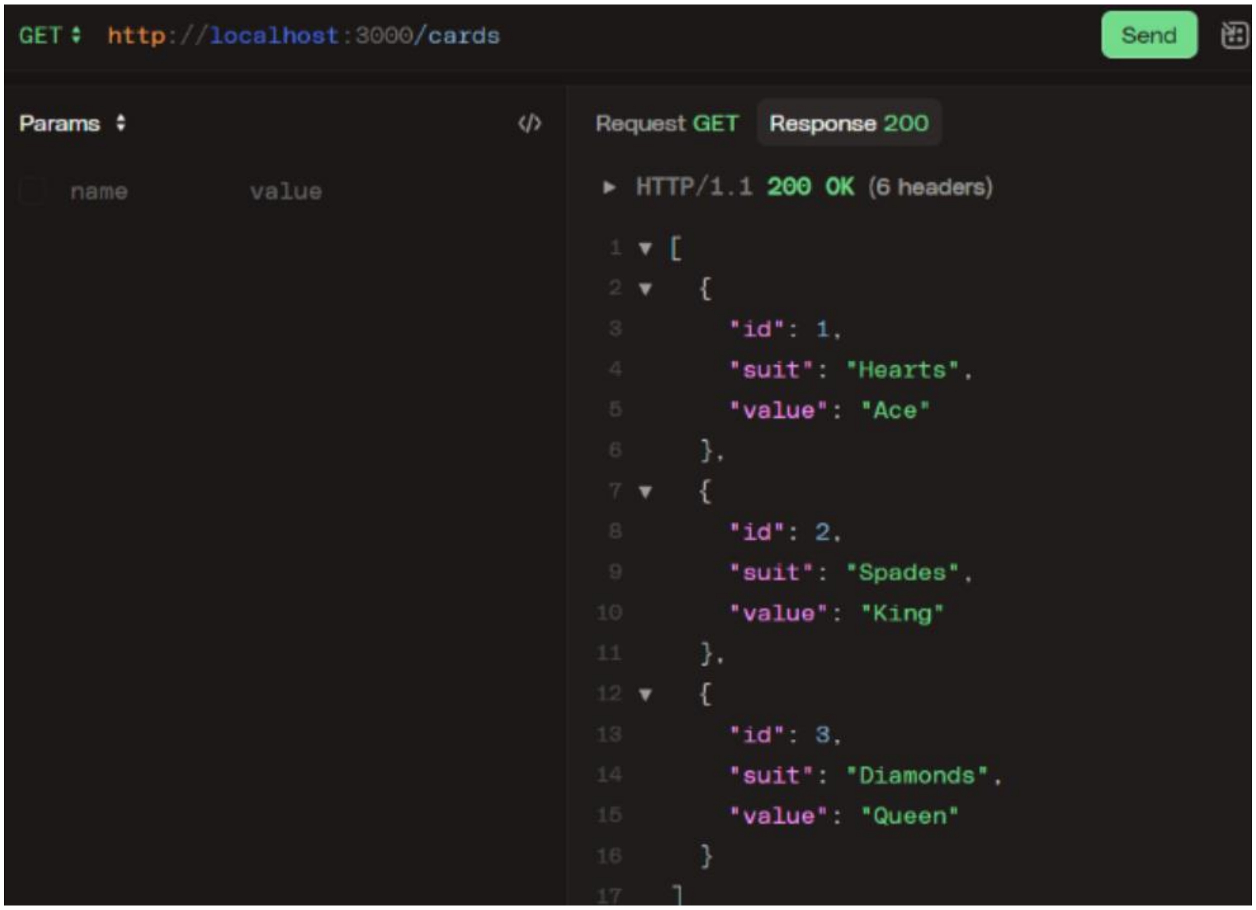
```
  res.status(201).json(newCard);
});

app.put('/api/cards/:id', (req, res) => {
  const card = cards.find(c => c.id === parseInt(req.params.id));
  if (!card) return res.status(404).json({ error: 'Card not found' });
  const { suit, value, collection } = req.body;
  if (suit) card.suit = suit;
  if (value) card.value = value;
  if (collection) card.collection = collection;
  res.json(card);
});

app.delete('/api/cards/:id', (req, res) => {
  const index = cards.findIndex(c => c.id === parseInt(req.params.id));
  if (index === -1) return res.status(404).json({ error: 'Card not found' });
  cards.splice(index, 1);
  res.status(204).send();
});

// server
const PORT =3000;
app.listen(PORT, () => console.log(Server running on port ${PORT}));
```

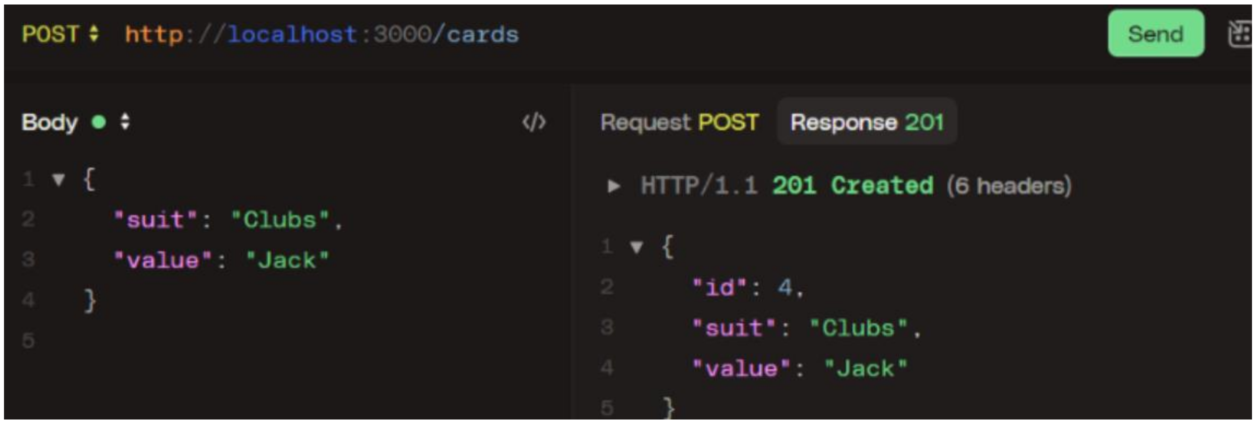
OUTPUT:



```
GET http://localhost:3000/cards
```

name value

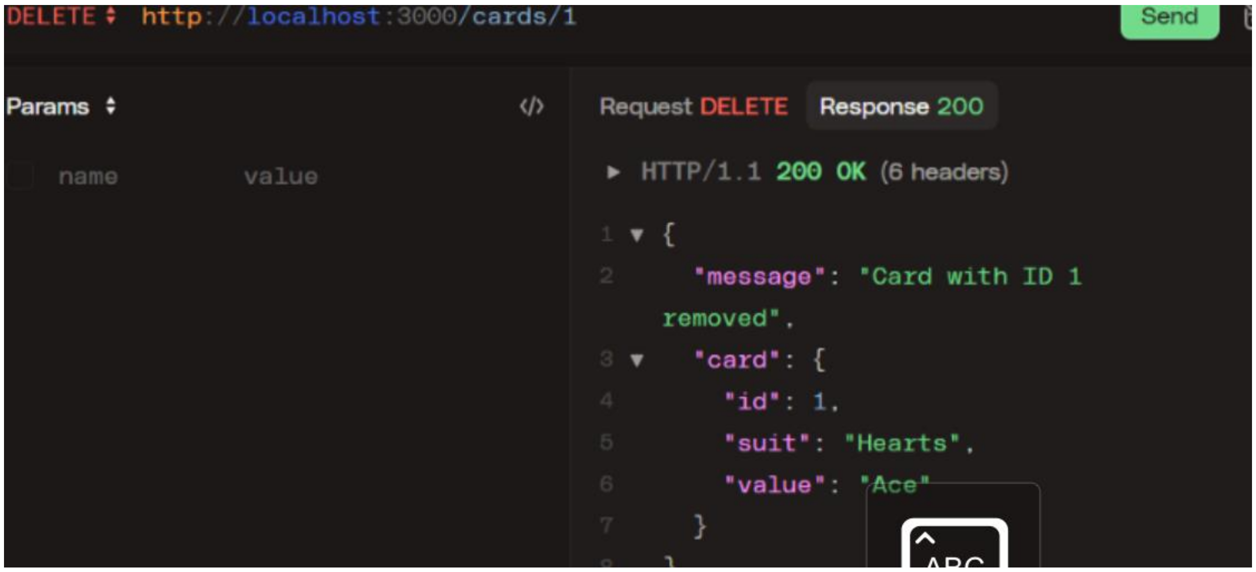
```
HTTP/1.1 200 OK (6 headers)
[
  {
    "id": 1,
    "suit": "Hearts",
    "value": "Ace"
  },
  {
    "id": 2,
    "suit": "Spades",
    "value": "King"
  },
  {
    "id": 3,
    "suit": "Diamonds",
    "value": "Queen"
  }
]
```



```
POST http://localhost:3000/cards
```

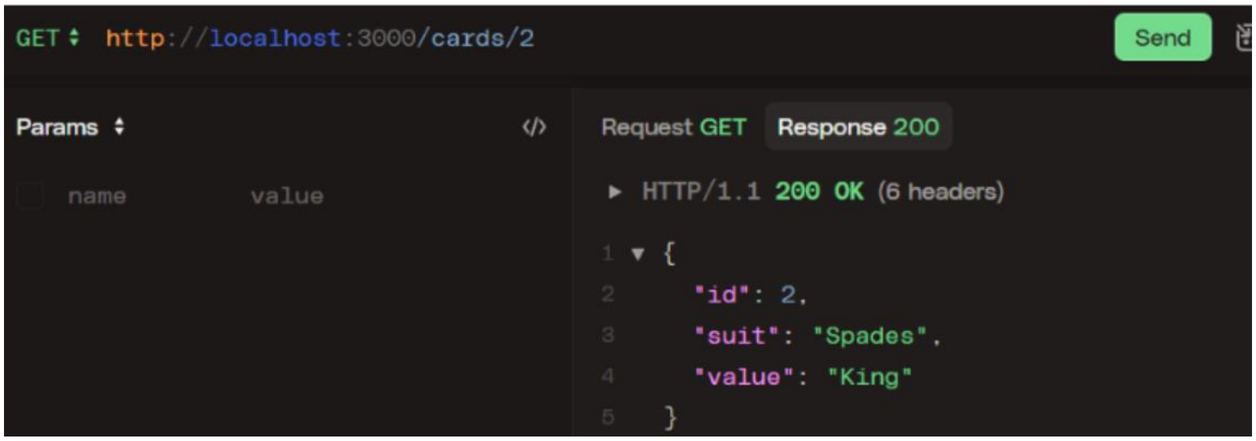
```
{
  "suit": "Clubs",
  "value": "Jack"
}
```

```
HTTP/1.1 201 Created (6 headers)
{
  "id": 4,
  "suit": "Clubs",
  "value": "Jack"
}
```



```
DELETE http://localhost:3000/cards/1
```

```
HTTP/1.1 200 OK (6 headers)
{
  "message": "Card with ID 1
removed",
  "card": {
    "id": 1,
    "suit": "Hearts",
    "value": "Ace"
  }
```



```
GET http://localhost:3000/cards/2
```

```
HTTP/1.1 200 OK (6 headers)
{
  "id": 2,
  "suit": "Spades",
  "value": "King"
}
```