

# Backend Project Exercise

## Project #1. Student Profile Finder:

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

// Create an Express application and body-parser

const app = express();

const PORT = 3000;

// Sample data: student array

let students = [

  { id: 0, name: 'unknown', course: 'unknown' },

  { id: 1, name: 'Neha Joshi', course: 'Data Science' },

  { id: 2, name: 'Tarun Kumar', course: 'Full Stack' },

  { id: 3, name: 'Manisha Sharma', course: 'AI & ML' },

  { id: 4, name: 'Utkarsh Mittal', course: 'UI/UX Design' },

  { id: 5, name: 'Ravi Mehta', course: 'Cybersecurity' },

  { id: 6, name: 'Sneha Roy', course: 'Cloud Computing' },

  { id: 7, name: 'Kunal Verma', course: 'DevOps' },

  { id: 8, name: 'Pooja Singh', course: 'Frontend Development' },

  { id: 9, name: 'Aman Kapoor', course: 'Backend Development' },
```

```
    { id: 10, name: 'Anjali Desai', course: 'Blockchain Technology' }

];

// Routes

// GET /students - Get all students

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

    res.json(employees);

});

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

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

    const student = students.find(emp => emp.id === empId);

    if (student) {

        res.json(student);

    } else {

        res.status(404).json({ message: 'Student not found - 404 error' });

    }

});

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

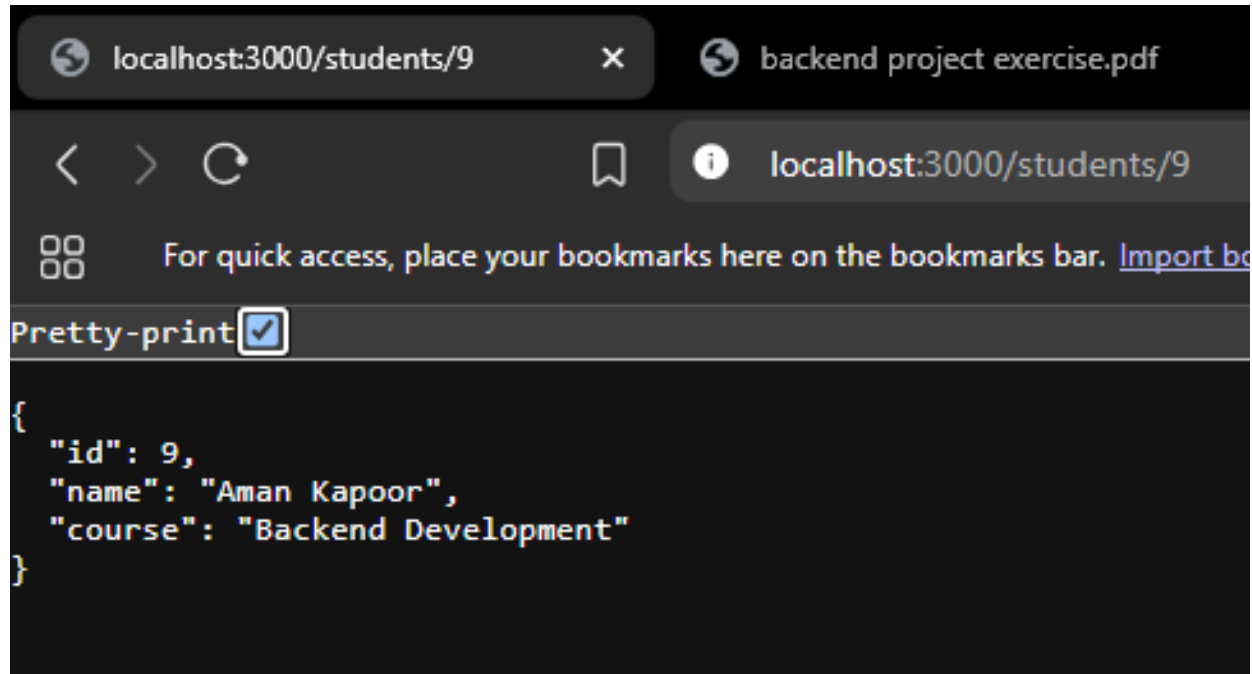
    res.send('Go To /students/:id Route')

});
```

```
// Start the server

app.listen(PORT, ()=>{console.log('Server is running...')});
```

## Output:



## Project #2. Course Search Engine:

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

const app = express();

app.use(express.json());

const PORT = 3000;

let students = [

  { id: 0, name: 'unknown', course: 'unknown' },

  { id: 1, name: 'Neha Joshi', course: 'Data Science' },

  { id: 2, name: 'Tarun Kumar', course: 'Full Stack' },
```

```
{ id: 3, name: 'Manisha Sharma', course: 'AI & ML' },
{ id: 4, name: 'Utkarsh Mittal', course: 'UI/UX Design' },
{ id: 5, name: 'Ravi Mehta', course: 'Cybersecurity' },
{ id: 6, name: 'Sneha Roy', course: 'Cloud Computing' },
{ id: 7, name: 'Kunal Verma', course: 'DevOps' },
{ id: 8, name: 'Pooja Singh', course: 'Frontend Development' },
{ id: 9, name: 'Aman Kapoor', course: 'Backend Development' },
{ id: 10, name: 'Anjali Desai', course: 'Blockchain Technology' }
];

// GET /students (with optional course filter)
app.get('/students', (req, res) => {
  const { course } = req.query;

  if (course) {
    const filtered = students.filter(s =>
      s.course.toLowerCase() === course.toLowerCase()
    );

    if(filtered.length > 0)
      return res.json(filtered)

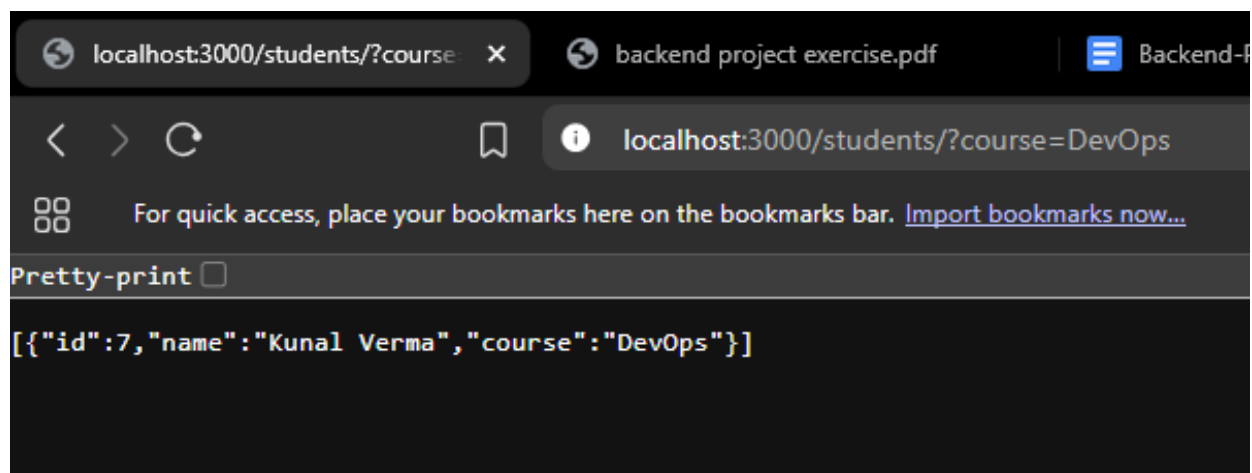
    else res.status(404).json({ message: 'No students found for this
course' });
  }

  res.json(students);
});

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

```
    res.send('Go To /students or /students/:id');
  });
app.listen(PORT, () => {
  console.log(`Server is running...`);
});
```

## Output:



## Project #3. Admission Form Handler:

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

const app = express();

app.use(express.json());

const PORT = 3000;

let students = [

  { id: 0, name: 'unknown', course: 'unknown', email:
'unknown@example.com' },

  { id: 1, name: 'Neha Joshi', course: 'Data Science', email:
'neha.joshi@example.com' },
```

```
    { id: 2, name: 'Tarun Kumar', course: 'Full Stack', email:
'tarun.kumar@example.com' },

    { id: 3, name: 'Manisha Sharma', course: 'AI & ML', email:
'manisha.sharma@example.com' },

    { id: 4, name: 'Utkarsh Mittal', course: 'UI/UX Design', email:
'utkarsh.mittal@example.com' },

    { id: 5, name: 'Ravi Mehta', course: 'Cybersecurity', email:
'ravi.mehta@example.com' },

    { id: 6, name: 'Sneha Roy', course: 'Cloud Computing', email:
'sneha.roy@example.com' },

    { id: 7, name: 'Kunal Verma', course: 'DevOps', email:
'kunal.verma@example.com' },

    { id: 8, name: 'Pooja Singh', course: 'Frontend Development', email:
'pooja.singh@example.com' },

    { id: 9, name: 'Aman Kapoor', course: 'Backend Development', email:
'aman.kapoor@example.com' },

    { id: 10, name: 'Anjali Desai', course: 'Blockchain Technology', email:
'anjali.desai@example.com' }
];

// POST /students

app.post('/students/admission', (req, res) => {

  const { id, name, course, email, } = req.body;

  if (!id || !name || !course || !email) {

    return res.status(400).json({ message: 'Missing fields: id, name, or
course' });

  }

  if (students.find(student => student.id === id)) {

    return res.status(409).json({ message: 'Student with this ID already
exists' });

  }

});
```

```
}

students.push({ id, name, course, email });

res.status(201).send('201 Created - Thank you Shubham for registering in Node.js');

});

// Root

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

  res.send('Go To /students or /students/:id');

});

app.listen(PORT, () => {

  console.log(`Server is running...`);

});
```

## Output:

The screenshot shows a web browser interface for a REST client. The URL bar displays `http://localhost:3000/students/admission`. The request method is set to **POST**. The request body is a JSON object:

```
{
  "id": 11,
  "name": "Shubham",
  "course": "Node.js",
  "email": "Shubham@example.com"
}
```

The response status is **201 Created**, with a response time of 38 ms and a body size of 291 B. The response body is displayed as:

```
1 201 Created - Thank you Shubham for registering in Node.js
```

## Project #4. Institute Info API:

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

const app = express();

const PORT = 3000;

// Middleware to log each request method and URL

app.use((req, res, next) => {

  console.log(`${req.method} ${req.url}`);

  next();

});

// Static route: /about

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

  res.send(`

    <h1>About TechBridge Institute</h1>

    <p>TechBridge Institute was founded in 2018 with the goal of bridging
the gap between academic learning and industry skills.

    With over 3,000 students graduated and 92% placement success, we offer
hands-on training with real-world projects and mentorship

    from IT professionals across the globe.</p>

  `);

});
```



```
// Static route: /contact

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

  res.send(`

    <h1>Contact Us</h1>

    <p>Email: support@techbridge.edu.in</p>

    <p>Phone: +91 98765 43210</p>

    <p>Address: 5th Floor, Sigma Tech Park, Bengaluru - 560103</p>

  `);

});

// Static route: /services

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

  res.send(`

    <h1>Our Services</h1>

    <ul>

      <li>Full Stack Development Bootcamp</li>

      <li>AI & Machine Learning Program</li>

      <li>Cybersecurity Essentials</li>

      <li>Cloud DevOps Certification</li>

      <li>UI/UX Design Sprint Course</li>

    </ul>

  `);

});
```

```
</ul>

`);

});

// Home route

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

    res.send('Welcome to TechBridge Institute API. Try /about, /contact, or /services');

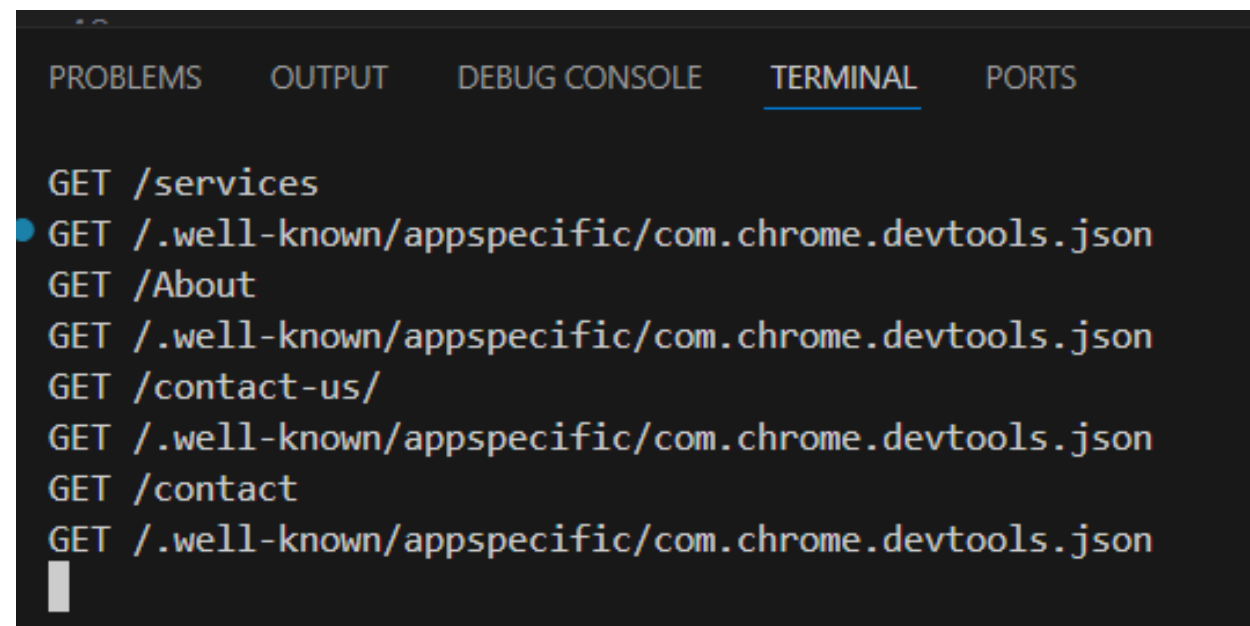
});

app.listen(PORT, () => {

    console.log(`Server is running...`);

});
```

## Output:



The screenshot shows a web browser's developer tools terminal. The 'TERMINAL' tab is selected, displaying a series of GET requests. The requests are: GET /services, GET /.well-known/appspecific/com.chrome.devtools.json, GET /About, GET /.well-known/appspecific/com.chrome.devtools.json, GET /contact-us/, GET /.well-known/appspecific/com.chrome.devtools.json, GET /contact, and GET /.well-known/appspecific/com.chrome.devtools.json. A cursor is visible at the end of the last line.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

GET /services
GET /.well-known/appspecific/com.chrome.devtools.json
GET /About
GET /.well-known/appspecific/com.chrome.devtools.json
GET /contact-us/
GET /.well-known/appspecific/com.chrome.devtools.json
GET /contact
GET /.well-known/appspecific/com.chrome.devtools.json
```

## Project #5. Route Tester with Error Pages:

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

const app = express();

const PORT = 3000;

// Middleware to log all requests
app.use((req, res, next) => {

  console.log(`${req.method} ${req.url}`);

  next();

});

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

  res.send('Welcome to the Home Page!');

});

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

  res.send('This project demonstrates route testing with error handling.');
```

```
});

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

  const error = new Error('Something went wrong internally.');
```

```
    error.status = 500;

    next(error);
  });

  // (catch-all for undefined routes)
  app.use((req, res, next) => {

    res.status(404).send('404 Page Not Found');

  });

  app.use((err, req, res, next) => {

    console.error('Internal Error:', err.message);

    res.status(err.status || 500).send('500 Internal Server Error');

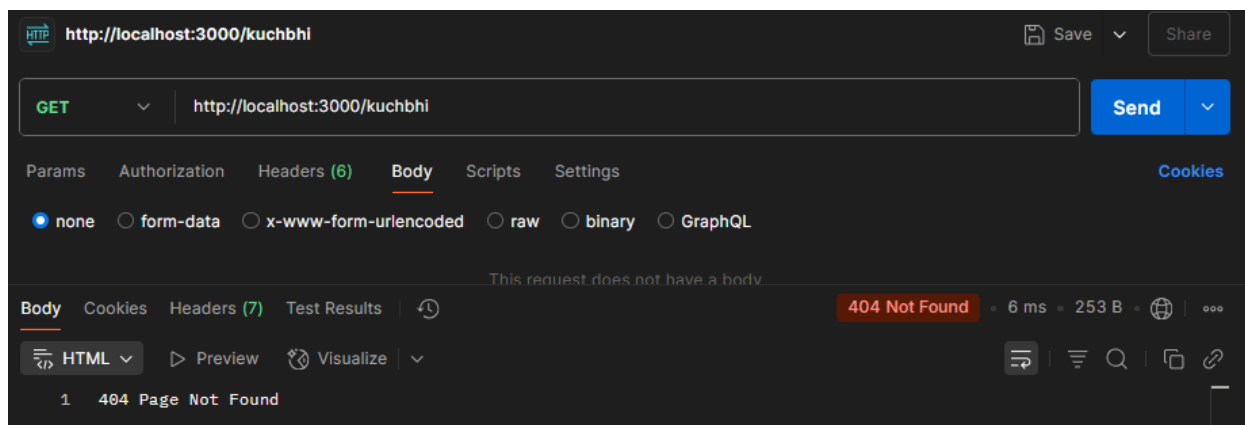
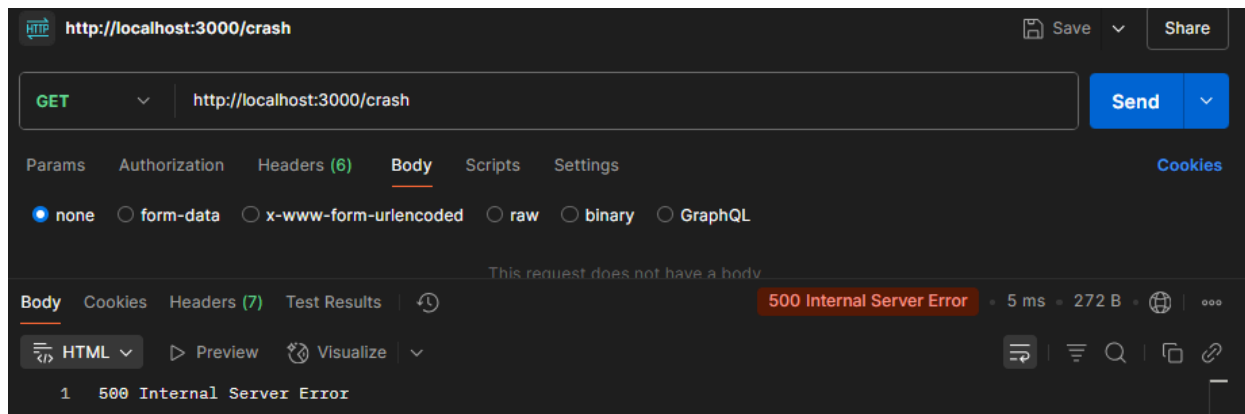
  });

  // Start the server
  app.listen(PORT, () => {

    console.log(`Server is running...`);

  });
```

## Output:



## Project #6. Feedback Collector:

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

const app = express();

const PORT = 3000;

app.use(express.json());

// POST /feedback route

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

  const { name, email, message } = req.body;
```

```
// Validate required fields

if (!name || !email || !message) {

    return res.status(400).json({ error: 'All fields (name, email, message) are required.' });

}

// Success response

res.send(`Thank you for your feedback, ${name}!`);

});

// Start the server

app.listen(PORT, () => {

    console.log(`Server is running...`);

});
```

## Output:

The screenshot shows a web browser interface for testing an API endpoint. The URL bar shows `http://localhost:3000/feedback`. The request method is `POST`. The request body is a JSON object:

```
{
  "name": "Shubham",
  "email": "shubham@email.com",
  "message": "We are learning MERN Stack here."
}
```

The response status is `200 OK` with a response time of `22 ms` and a size of `265 B`. The response body is `Thank you for your feedback, Shubham!`.

## Project #7. Course Details API:

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

const app = express();

const PORT = 3000;

// Mock course list

const courses = [

  { id: 1, title: 'Web Development Basics', duration: '2 Months' },

  { id: 2, title: 'Frontend with React', duration: '1.5 Months' },

  { id: 3, title: 'Backend with Express', duration: '1 Month' },

  { id: 4, title: 'Full Stack Bootcamp', duration: '3 Months' },

  { id: 5, title: 'Advanced Node.js', duration: '1 Month' },

  { id: 6, title: 'Python for Data Science', duration: '2 Months' }

];

// GET /courses/:courseId

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

  const courseId = parseInt(req.params.courseId);

  const course = courses.find(course => course.id === courseId);

  if (course) {

    res.json(course);

  } else {

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

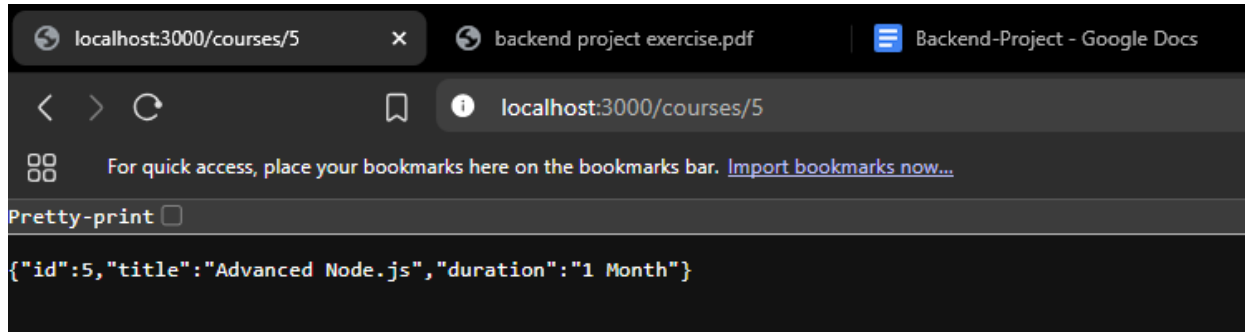
  }

});

// Start server
```

```
app.listen(PORT, () => {  
  
  console.log(`Server is running...`);  
  
});
```

## Output:



## Project #8. Simple Blog Viewer:

```
const express = require('express');  
  
const app = express();  
  
const PORT = 3000;  
  
const blogs = [  
  
  { id: 1, title: 'Intro to Node.js', content: 'Node.js is a JavaScript runtime built on Chrome\'s V8 engine...' },  
  
  { id: 2, title: 'Routing in Express', content: 'Routing in Express helps you define endpoints for your app...' },  
  
  { id: 3, title: 'Middleware Explained', content: 'Middleware functions have access to req, res, and next...' }  
  
];  
  
// Middleware to log visits to any /blogs route  
  
app.use('/blogs', (req, res, next) => {
```



```
    console.log(`BLOG ACCESS: ${req.method} ${req.url}`);

    next();
  });

  // GET /blogs → list of blog titles
  app.get('/blogs', (req, res) => {

    const titles = blogs.map(blog => blog.title);

    res.json(titles);
  });

  // GET /blogs/:id → full blog post
  app.get('/blogs/:id', (req, res) => {

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

    const blog = blogs.find(b => b.id === blogId);

    if (blog) {

      res.json(blog);

    } else {

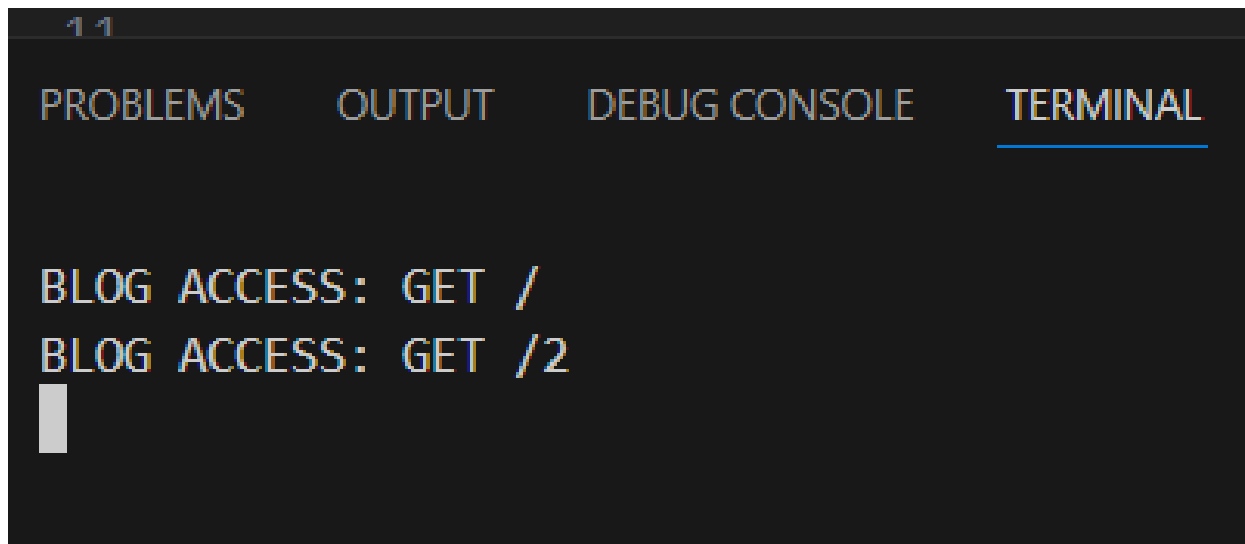
      res.status(404).json({ message: 'Blog not found' });

    }
  });

  // Start server
  app.listen(PORT, () => {

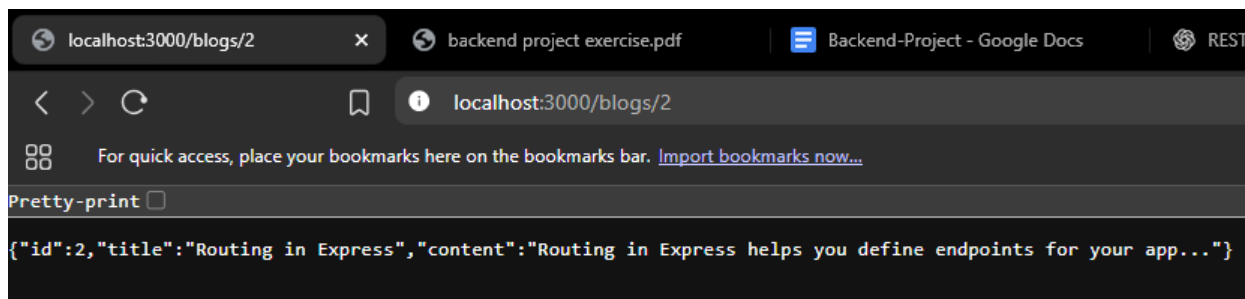
    console.log(`Server is running...`);
  });
```

## Output:



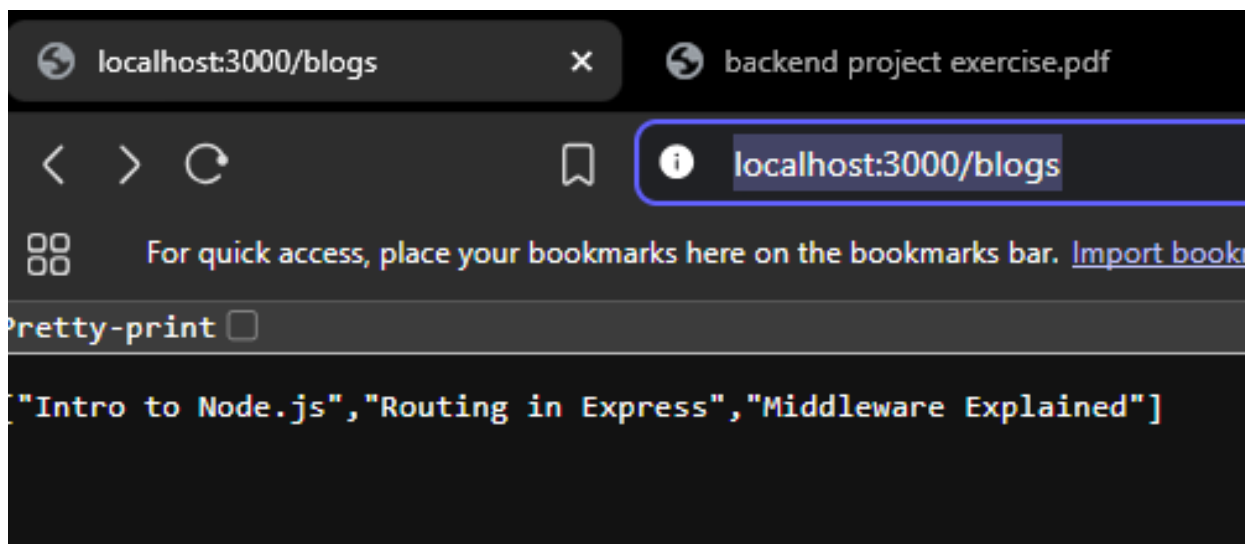
A screenshot of the Visual Studio Code interface with the 'TERMINAL' tab selected. The terminal displays two log entries: 'BLOG ACCESS: GET /' followed by 'BLOG ACCESS: GET /2'. A cursor is visible on the line following the second log entry.

```
1 1  
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL  
  
BLOG ACCESS: GET /  
BLOG ACCESS: GET /2  
|
```



A screenshot of a web browser window showing the URL 'localhost:3000/blogs/2'. The page content displays a JSON object representing a blog entry.

```
localhost:3000/blogs/2  backend project exercise.pdf  Backend-Project - Google Docs  REST  
< > ↻  localhost:3000/blogs/2  
For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...  
Pretty-print ☐  
{ "id": 2, "title": "Routing in Express", "content": "Routing in Express helps you define endpoints for your app..." }
```



A screenshot of a web browser window showing the URL 'localhost:3000/blogs'. The page content displays an array of blog titles.

```
localhost:3000/blogs  backend project exercise.pdf  
< > ↻  localhost:3000/blogs  
For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...  
Pretty-print ☐  
[ "Intro to Node.js", "Routing in Express", "Middleware Explained" ]
```

**END**