Practical No. 7: Study And Implementation of Express.js

Problem Statement 1: Basics of Express.js

What is Express.js and how does it differ from Node.js?

Express.js is a minimal and flexible web application framework built on top of Node.js. While Node.js provides the core runtime environment and low-level APIs to handle HTTP requests, Express.js offers a simpler interface and a robust set of features like routing, middleware support, and template engines for building web applications and APIs more efficiently.

How do you create a simple Express.js server?

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

const app = express();


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

  res.send('Hello, Express!');

});


app.listen(3000, () => {

  console.log('Server running on http://localhost:3000');

});
```

Explain the concept of routing in Express.js. How do you define routes?

Routing in Express.js refers to how an application responds to client requests at specific endpoints (URLs) using specific HTTP methods. Routes are defined using methods like app.get(), app.post(), etc.

Example:

```
app.get('/about', (req, res) => {
  res.send('About Page');
});
```

What is middleware in Express.js, and how does it work?

Middleware in Express.js is a function that has access to the req, res, and next objects. It can execute code, modify the request/response objects, end the request-response cycle, or call the next middleware function. Middleware functions are executed in sequence.

How do you create and use custom middleware in an Express.js application?

```
const logMiddleware = (req, res, next) => {
  console.log(`${req.method} ${req.url}`);
  next();
};


app.use(logMiddleware);
```

This middleware logs each request's method and URL before passing control to the next handler.

What is the difference between application-level middleware and router-level middleware?

- **Application-level middleware** is bound to an instance of express() and applies to all routes.
- app.use((req, res, next) => { ... });
- **Router-level middleware** is bound to an instance of express.Router() and applies only to that router.
- const router = express.Router();
- router.use((req, res, next) => { ... });

What are req and res in Express.js? Give examples of common properties and methods associated with each.

- req (Request object) contains information about the HTTP request.
    - Common properties: req.params, req.query, req.body, req.headers
- res (Response object) is used to send a response to the client.
    - Common methods: res.send(), res.json(), res.status(), res.redirect()

How would you extract query parameters from a URL in an Express.js route?

Query parameters can be accessed using req.query.

Example:

```
app.get('/search', (req, res) => {
  const keyword = req.query.q;
  res.send(`Searching for ${keyword}`);
});
```

How does Express.js handle different HTTP methods (GET, POST, PUT, DELETE)?

Express provides methods like app.get(), app.post(), app.put(), and app.delete() to handle different types of HTTP requests.

Example:

```
app.post('/create', (req, res) => {
  res.send('Created');
});
```

What are route parameters in Express.js? How do you use them in a route definition?

Route parameters are dynamic segments in the URL, prefixed with a colon (:). They are accessed using req.params.

Example:

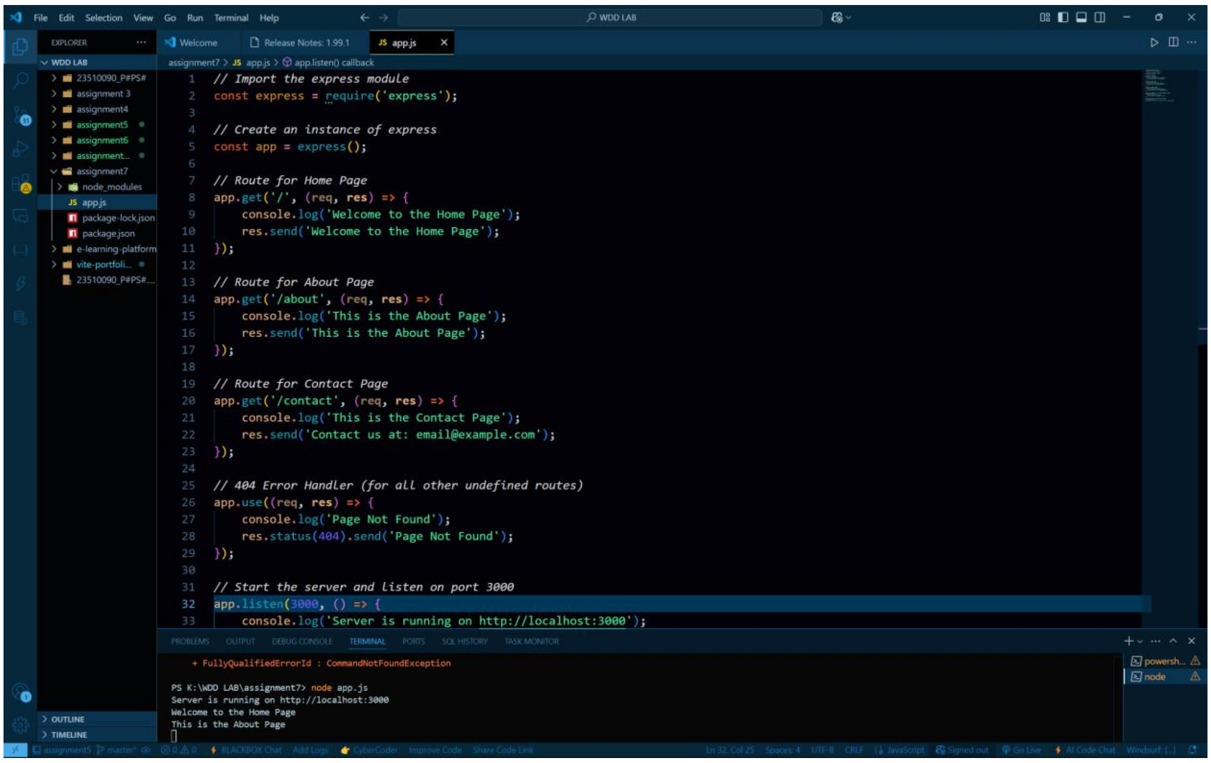
```
app.get('/user/:id', (req, res) => {
  res.send(`User ID is ${req.params.id}`);
});
```

Problem Statement 2: Basic Web Server with Express.js Requirements

Create a basic Express.js server that listens on port 3000.

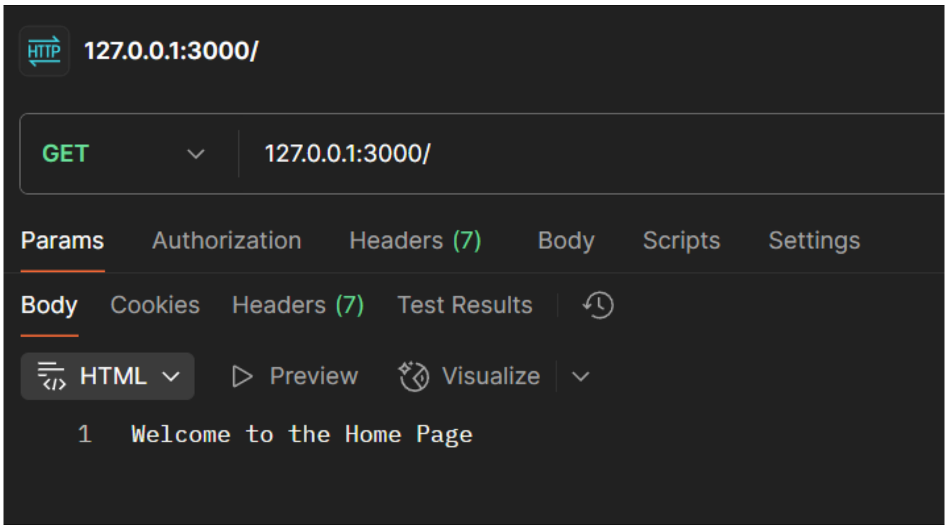
Define three routes:

- GET / - Responds with "Welcome to the Home Page".
- GET /about - Responds with "This is the About Page".
- GET /contact - Responds with "Contact us at: email@example.com".
- Include a 404 error handler that displays a "Page Not Found" message for unknown routes.


File Edit Selection View Go Run Terminal Help
WDD LAB
EXPLORER
Welcome
Release Notes: 1.99.1
app.js
assignment7 > app.js > app.listen() callback
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL HISTORY TASK MONITOR
OUTLINE
TIMELINE

```
// Import the express module
const express = require('express');

// Create an instance of express
const app = express();

// Route for Home Page
app.get('/', (req, res) => {
    console.log('Welcome to the Home Page');
    res.send('Welcome to the Home Page');
});

// Route for About Page
app.get('/about', (req, res) => {
    console.log('This is the About Page');
    res.send('This is the About Page');
});

// Route for Contact Page
app.get('/contact', (req, res) => {
    console.log('This is the Contact Page');
    res.send('Contact us at: email@example.com');
});

// 404 Error Handler (for all other undefined routes)
app.use((req, res) => {
    console.log('Page Not Found');
    res.status(404).send('Page Not Found');
});

// Start the server and listen on port 3000
app.listen(3000, () => {
    console.log('Server is running on http://localhost:3000');
```

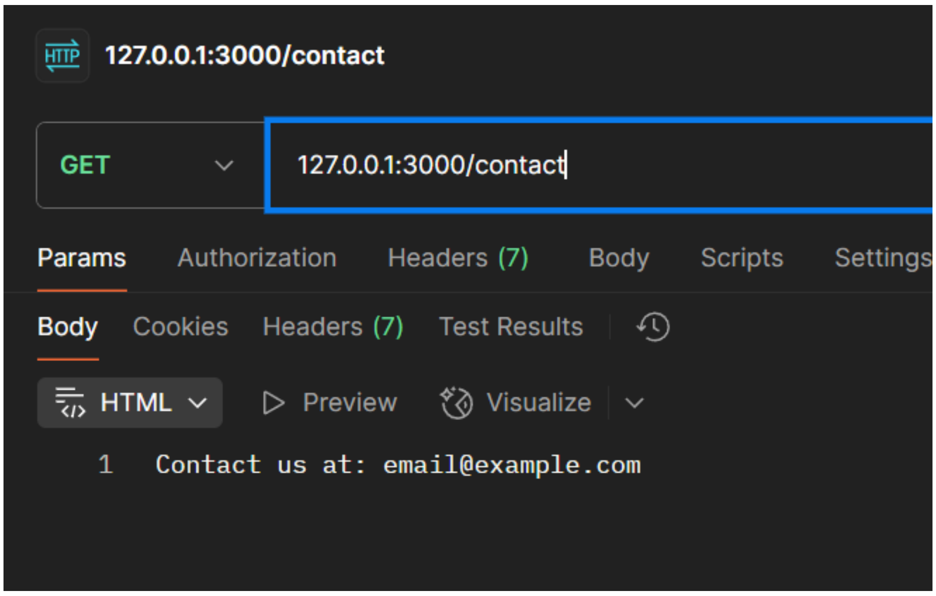
```
    + FullyQualifiedErrorId : CommandNotFoundException

PS K:\WDD LAB\assignment7> node app.js
Server is running on http://localhost:3000
Welcome to the Home Page
This is the About Page
```



127.0.0.1:3000/

GET 127.0.0.1:3000/

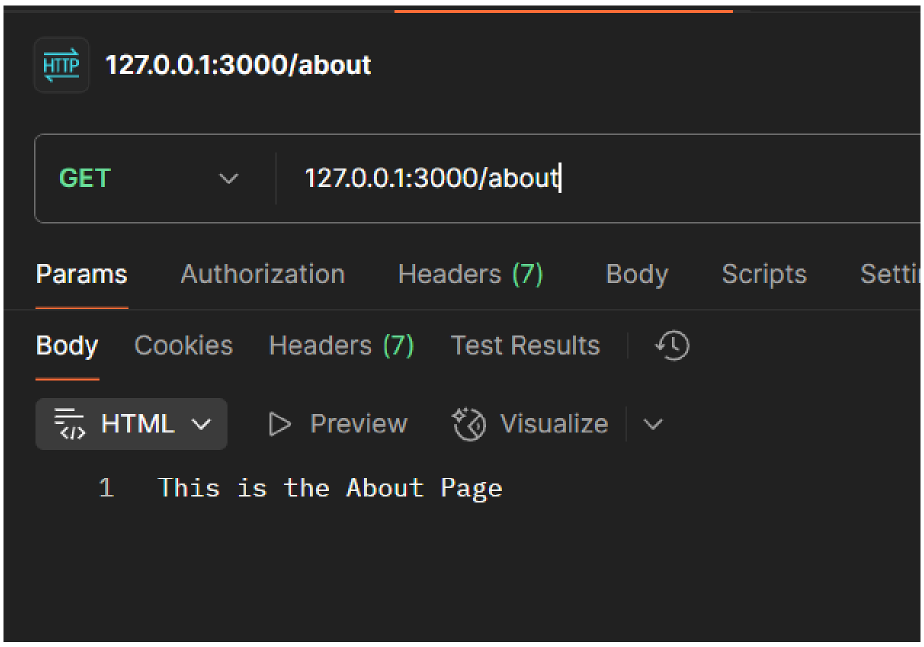
```
Welcome to the Home Page
```



127.0.0.1:3000/contact

GET 127.0.0.1:3000/contact

```
Contact us at: email@example.com
```



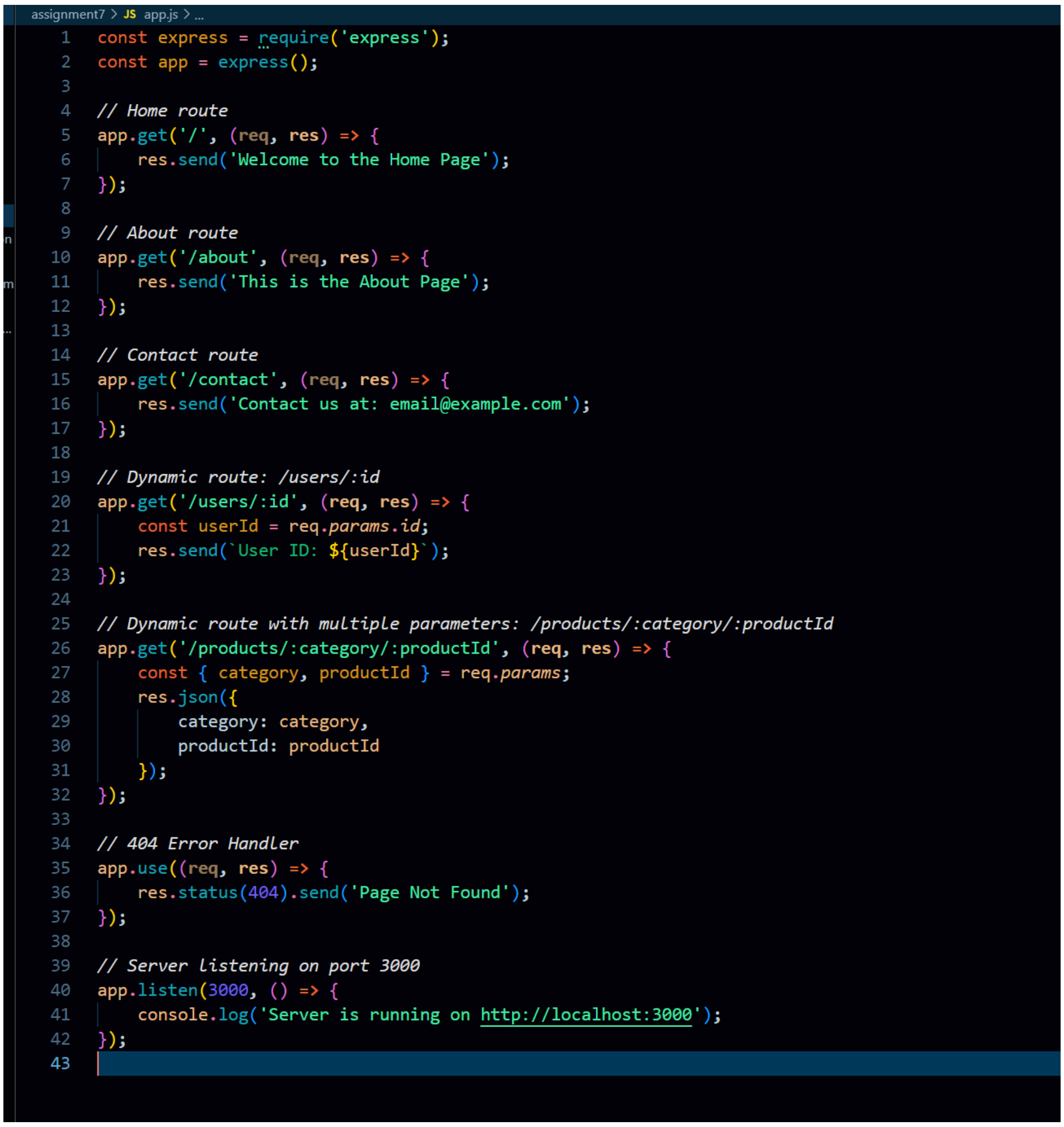
127.0.0.1:3000/about

GET 127.0.0.1:3000/about

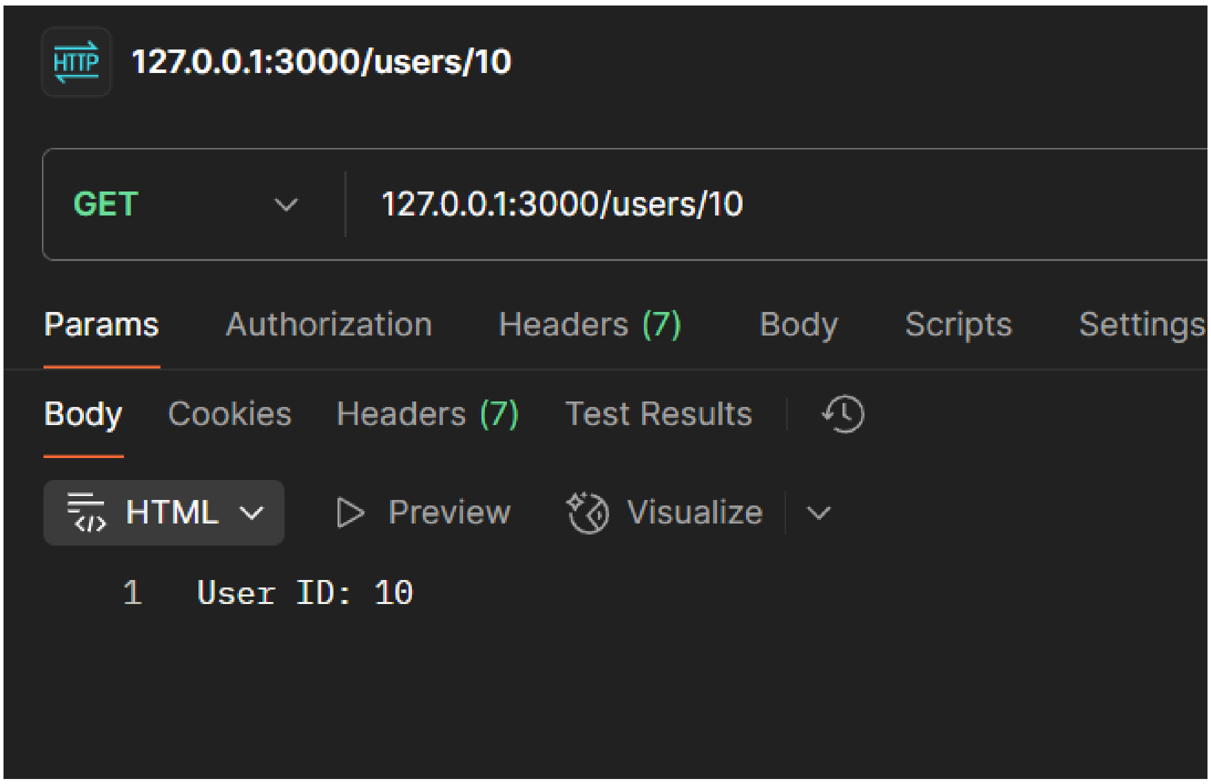
```
This is the About Page
```

Problem Statement 3: Dynamic Route Parameters

- **Modify the previous server to include the following route:**
- **GET /users/:id - Responds with "User ID: [id]" where [id] is the dynamic value from the route.**
- **Add another route:**
  - **o GET /products/:category/:productId - Responds with "Category: [category], Product ID: [productId]".**
- **Return a JSON object containing the category and product ID instead of a plain string.Note:**

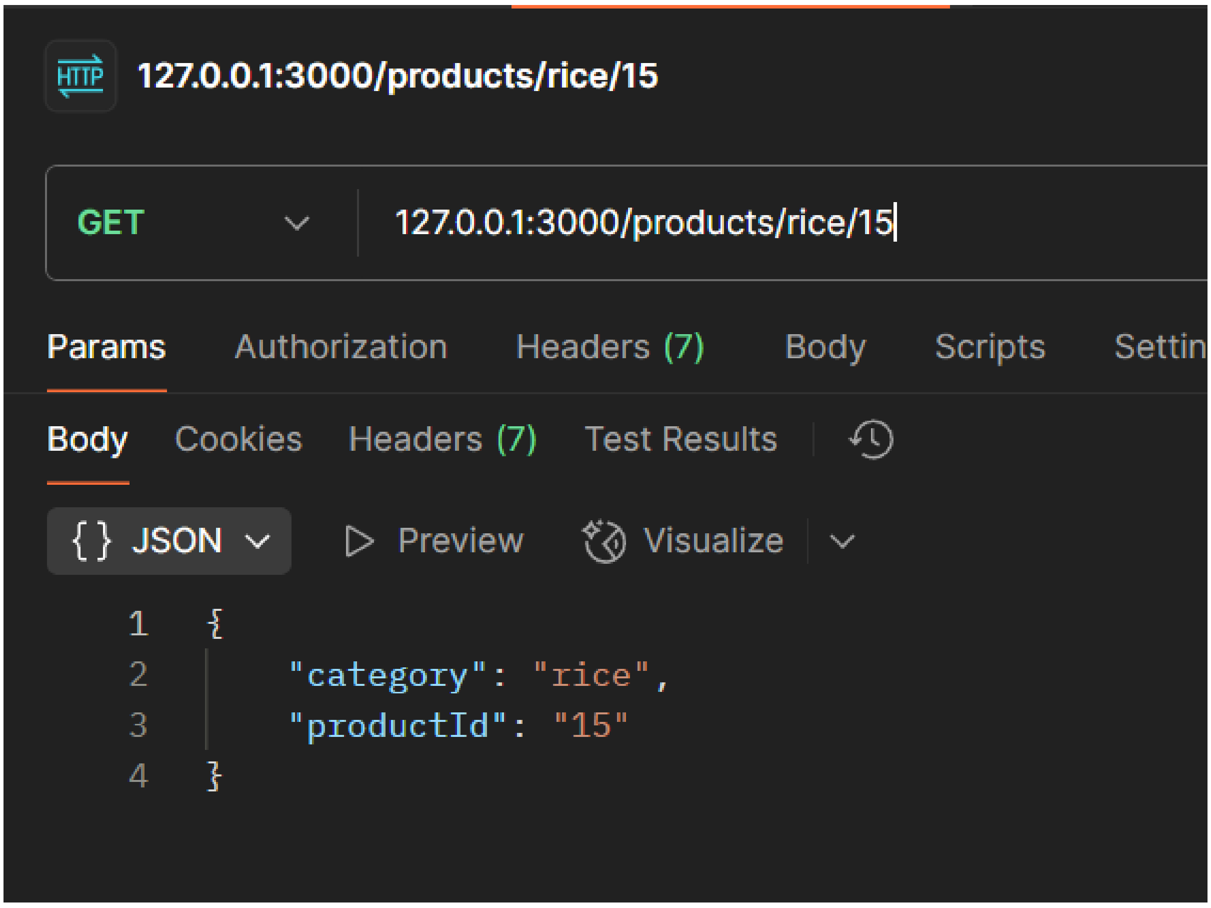


```javascript
const express = require('express');
const app = express();

// Home route
app.get('/', (req, res) => {
    res.send('Welcome to the Home Page');
});

// About route
app.get('/about', (req, res) => {
    res.send('This is the About Page');
});

// Contact route
app.get('/contact', (req, res) => {
    res.send('Contact us at: email@example.com');
});

// Dynamic route: /users/:id
app.get('/users/:id', (req, res) => {
    const userId = req.params.id;
    res.send(`User ID: ${userId}`);
});

// Dynamic route with multiple parameters: /products/:category/:productId
app.get('/products/:category/:productId', (req, res) => {
    const { category, productId } = req.params;
    res.json({
        category: category,
        productId: productId
    });
});

// 404 Error Handler
app.use((req, res) => {
    res.status(404).send('Page Not Found');
});

// Server listening on port 3000
app.listen(3000, () => {
    console.log('Server is running on http://localhost:3000');
});

```


127.0.0.1:3000/users/10

GET 127.0.0.1:3000/users/10

```
User ID: 10
```


127.0.0.1:3000/products/rice/15

GET 127.0.0.1:3000/products/rice/15

```
{
    "category": "rice",
    "productId": "15"
}
```