To set up a simple Node.js server that connects to a MySQL database and exposes the required endpoints, follow these steps:

1. Setup the Environment

First, ensure you have express, mysql2, and dotenv installed. These are necessary for setting up the server, interacting with MySQL, and using environment variables.

Run the following command to install them:

Npm install express mysql2 dotenv

2. Create the .env file

In the root of your project, create a .env file to store your database credentials securely.

DB_HOST=localhost

DB_USER=root

DB_PASSWORD=your_password

DB_NAME=your_database_name

Make sure to replace your_password and your_database_name with your actual MySQL credentials.

3. Create server.js

Now, create a server.js file where we will define the database connection and routes.

// Import dependencies

```
Const express = require(‘express’);
Const mysql = require(‘mysql2’);
Const dotenv = require(‘dotenv’);

// Load environment variables from .env
Dotenv.config();

// Create an Express app
Const app = express();

// Set the port
Const port = 3000;

// Create MySQL connection
Const connection = mysql.createConnection({
  Host: process.env.DB_HOST,
  User: process.env.DB_USER,
  Password: process.env.DB_PASSWORD,
  Database: process.env.DB_NAME,
});

// Connect to the database
Connection.connect((err) => {
  If (err) {
    Console.error(‘Error connecting to the database: ‘ + err.stack);
    Return;
  }
  Console.log(‘Connected to the database’);
```

```
});

// Define routes

// Retrieve all patients
App.get('/patients', (req, res) => {
  Const query = 'SELECT patient_id, first_name, last_name, date_of_birth FROM
patients';
  Connection.query(query, (err, results) => {
    If (err) {
      Res.status(500).json({ message: 'Error retrieving patients', error: err });
      Return;
    }
    Res.json(results);
  });
});

// Retrieve all providers
App.get('/providers', (req, res) => {
  Const query = 'SELECT first_name, last_name, provider_specialty FROM providers';
  Connection.query(query, (err, results) => {
    If (err) {
      Res.status(500).json({ message: 'Error retrieving providers', error: err });
      Return;
    }
    Res.json(results);
  });
});
```

```
// Filter patients by first name
App.get('/patients/:first_name', (req, res) => {
  Const { first_name } = req.params;
  Const query = 'SELECT patient_id, first_name, last_name, date_of_birth FROM patients WHERE first_name = ?';
  Connection.query(query, [first_name], (err, results) => {
    If (err) {
      Res.status(500).json({ message: 'Error retrieving patients by first name', error: err });
      Return;
    }
    Res.json(results);
  });
});

// Retrieve providers by specialty
App.get('/providers/specialty/:specialty', (req, res) => {
  Const { specialty } = req.params;
  Const query = 'SELECT first_name, last_name, provider_specialty FROM providers WHERE provider_specialty = ?';
  Connection.query(query, [specialty], (err, results) => {
    If (err) {
      Res.status(500).json({ message: 'Error retrieving providers by specialty', error: err });
      Return;
    }
    Res.json(results);
  });
});
```

```
// Start the server
App.listen(port, () => {
  Console.log(`Server is running on port ${port}`);
});
```

Explanation:

Environment Variables: We use dotenv to load database credentials from the .env file. This keeps sensitive information like database credentials out of the codebase.

Database Connection: We use mysql2 to connect to the MySQL database using credentials from the environment variables.

GET Endpoints:

/patients: Retrieves all patients with their patient_id, first_name, last_name, and date_of_birth.

/providers: Retrieves all providers with their first_name, last_name, and provider_specialty.

/patients/:first_name: Filters patients by their first name.

/providers/specialty/:specialty: Filters providers by their specialty.

4. Test the Application

To run the application, use the following command:

Node server.js

Your application will start on http://localhost:3000.

You can test the endpoints using tools like Postman or by visiting the URLs in your browser:

Retrieve all patients: GET http://localhost:3000/patients

Retrieve all providers: GET http://localhost:3000/providers

Filter patients by first name: GET http://localhost:3000/patients/John

Retrieve providers by specialty: GET http://localhost:3000/providers/specialty/Pediatrics

Conclusion

This basic setup demonstrates how to configure a Node.js application to connect to a MySQL database and implement various endpoints for retrieving and filtering data. Make sure your MySQL server is running and the database credentials are correct in the .env file.