Steps to create a new Node Js project and basic setup for handling the Client HTTP requests.

1. Create a new folder and go to the created folder directory

2. Initialize a new Node.js project in cmd- explanation added after Step 5
   `npm init -y` or `npm init`

3. Install Required Packages

   ```
   npm install express
   ```

   ```
   npm install mysql
   ```

   ```
   npm install body-parser
   ```

4. Create a new file named index.js in your project directory

   ```
   const express = require('express');
   const mysql = require('mysql');
   const bodyParser = require('body-parser');
   const app = express();


   app.use(bodyParser.json());
   const port = 3001;
   const db = mysql.createConnection({
      host: 'localhost',
      user: 'root',
      password: 'root',
      database: ‘database_name_here’
   });
   db.connect();
   ```

```
app.get('/employees', (req, res) => {
   const sql = 'SELECT * FROM table_name_here;
   db.query(sql, (err, result) => {
       if (err) {
           console.log('Error fetching data:', err);
           res.status(500).send('Error fetching data');
       } else {
           console.log(result);
           res.status(200).json(result);
       }
   });
});

app.listen(port);
```

5. Start Node.js server in the terminal
   ```
   node index.js
   ```

Now, our Node.js server should be up and running and can access the /employees endpoint to fetch data from your MySQL database using port number 3001.
Ref: Client side request flow- https://www.yogeshrajadev.com

Explanation:

Step 2:
npm- managing packages and dependencies for Node.js applications.
Init- initializes a new npm package in the current directory. It creates a package.json file, which is used to manage the project's dependencies, metadata, and other configuration settings.
-y stands for “yes” or “yes to all”- accepts default values for all configuration settings without requiring any user input or confirmation.

Breakdown the Step 3:
```
const express = require('express');
```
Import the Express.js framework into the Node.js application
Express.js is a web application framework for Node.js, designed for building web applications and APIs.

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

imports the MySQL module, which is a Node.js driver for MySQL databases. It allows Node.js applications to interact with MySQL databases by providing methods for executing SQL queries and handling database connections.

```
const bodyParser = require('body-parser');
```

imports the body-parser middleware module. body-parser is used to parse the incoming request bodies in a middleware before your handlers, available under the req.body property.

```
const app = express();
```

It creates an instance of the Express application by calling the express() function. This app object represents the Express application and is used to configure routes, middleware, and other settings.

```
app.use(bodyParser.json());
```

the Express application to use the body-parser middleware for parsing JSON-formatted request bodies. It adds a middleware function to the Express request processing pipeline.

```
const port = 3001;
```

defines the port number on which the Express application will listen for incoming HTTP requests. In this case, the server will listen on port 3001.

```
const db = mysql.createConnection({
    host: 'host_name_here',
    user: 'user_name_here',
    password: 'password_here',
    database: 'database_name_here'
});
```

Creates a connection to the MySQL database. It uses the createConnection() method provided by the MySQL module to establish a connection to the MySQL server running on the local host with the specified username, password, and database name.

```
db.connect();
```

Establishes the connection to the MySQL database by calling the connect() method on the DB object. Once connected, the Node.js application can interact with the MySQL database.

```
app.get('/employees', (req, res) => {
    const sql = 'SELECT * FROM column_name_here;
    db.query(sql, (err, result) => {
        if (err) {
            console.log('Error fetching data:', err);
            res.status(500).send('Error fetching data');
        } else {
            console.log(result);
            res.status(200).json(result);
        }
    });
});
```
Defines a route handler for the GET requests to the /employees endpoint. When a GET request is received at this endpoint, it executes a SQL query to select all records from the column_name_here table in the MySQL database. The query result is then sent as a JSON response to the client.

```
app.listen(port)
```
It is necessary to start your Express server and make it listen for incoming HTTP requests. Without app.listen(), your Express application would not be able to accept and handle any incoming requests, rendering it non-functional as a web server.
Can you use like below to check whether the server starts or not?
```
app.listen(3001, function () {
    console.log('Server is listening on port', port);
});
```