Node-JS-API

Start with a concise overview of your Node.js API. Explain what the API does, its main features, and its purpose. This helps visitors quickly understand the project's scope and relevance.

To install and run your Node.js API project, follow these general steps assuming you have Node.js and npm (Node Package Manager) installed on your system:

Installation Steps

1. **Clone the Repository:** Clone your Node.js API project repository from wherever it is hosted (e.g., GitHub).

```
git clone <repository-url>
cd <project-directory>
```

2. **Install Dependencies:** Navigate to your project directory and install the required dependencies using npm.

   ```
   npm install
   ```

   This command will read the `package.json` file in your project directory and install all the dependencies listed under `dependencies` and `devDependencies`.

3. **Set Up Environment Variables (if applicable):** If your project uses environment variables for configuration (e.g., database connection strings, API keys), create a `.env` file in the root of your project and define your variables there. Make sure not to commit this file to version control by adding it to your `.gitignore`.

   Example `.env` file:

   ```
   PORT=3001
   DATABASE_URL=mongodb://localhost:27017/bookstore
   ```

4. **Start the Server:** Once dependencies are installed and environment variables are set (if needed), you can start your Node.js server.

   ```
   npm start
   ```

   This command will execute the script defined in the `start` field of your `package.json` file. Typically, it starts your Node.js application, which will listen for incoming requests on the specified port (e.g., 3001 in this case).

5. **Verify the Installation:** Open a web browser or use tools like Postman to send requests to your API endpoints to verify that it is working correctly.

Additional Notes

- **Database Setup:** If your API interacts with a database (e.g., MongoDB, MySQL), ensure the database server is running and that your API's connection configuration (like `DATABASE_URL` in `.env`) is correctly set.

- **Testing:** Consider writing and running tests for your API endpoints using frameworks like Mocha, Jest, or Postman's built-in testing features to ensure everything functions as expected.
- **Deployment:** For production deployment, consider using services like Heroku, AWS Elastic Beanstalk, or Azure App Service. Ensure your deployment environment is configured with appropriate security measures and scalability options.

Following these steps should help you successfully install and run your Node.js API project locally. Adjustments may be necessary based on specific project requirements and dependencies.

API Endpoints Documentation

Each section below describes a different endpoint available in the API.

Fetch all books

- **Description:** Retrieves all books available in the API.
- **Request:** GET http://localhost:3001/books

Fetch a specific book by ID

- **Description:** Retrieves a book from the API based on its ID.
- **Request:**

  ```
  GET http://localhost:3001/books/{id}
  ```

  Replace `{id}` with the ID of the book.

Fetch books by title

- **Description:** Retrieves books from the API based on their title.

- **Request:**

```
GET http://localhost:3001/booksTitle/{title}
```

Replace `{title}` with the title of the book.

Fetch all authors

- **Description:** Retrieves all authors from the API.
- **Request:**

```
GET http://localhost:3001/authors
```

Fetch a specific author by ID

- **Description:** Retrieves an author from the API based on their ID.
- **Request:**

```
GET http://localhost:3001/authors/{id}
```

Replace `{id}` with the ID of the author.

Fetch author's country by author ID

- **Description:** Retrieves the country of an author from the API based on their ID.
- **Request:**

```
GET http://localhost:3001/author/Country/{id}
```

Replace `{id}` with the ID of the author.

Fetch all customers

- **Description:** Retrieves all customers from the API.
- **Request:**

```
GET http://localhost:3001/customers
```

Fetch a specific customer by ID

- **Description:** Retrieves a customer from the API based on their ID.
- **Request:**

```
GET http://localhost:3001/customers/{id}
```

Replace `{id}` with the ID of the customer.

Fetch customer by phone number

- **Description:** Retrieves a customer from the API based on their phone number in E.164 format.
- **Request:**

```
GET http://localhost:3001/customer/Phone/{phoneNumber}
```

Replace `{phoneNumber}` with the customer's phone number in E.164 format.

Fetch all orders

- **Description:** Retrieves all orders from the API.
- **Request:**

```
GET http://localhost:3001/orders
```

Fetch a specific order by ID

- **Description:** Retrieves an order from the API based on its ID.
- **Request:**

```
GET http://localhost:3001/orders/{id}
```

Replace `{id}` with the ID of the order.

Fetch orders by date

- **Description:** Retrieves orders from the API based on a specific date.
- **Request:**

```
GET http://localhost:3001/ordersDate/{date}
```

Replace `{date}` with the date in 'YYYY-MM-DD' format.
