INNOMATİCS
RESEARCH LABS

To-Do List API Documentation

This task guides you in building a **To-Do List API** using Node.js and Express. The goal is to create an API that allows users to add, retrieve, update, and delete to-do list items. This will help you gain practical experience with HTTP methods and basic CRUD operations.

Prerequisites

- **Node.js** and **npm** installed on your machine.
- Familiarity with basic JavaScript, JSON, and HTTP methods.
- Create an Atlas account for MongoDB to create a DataBase.

1. Setting Up the Environment

1. **Initialize the Project**:
    - Run `npm init -y` to create a `package.json` file.
    - Install Express with `npm install express`.
2. **Create an Entry File**:
    - In the root directory, create an `index.js` file. This will serve as the main file to start the server.
3. **Starting the Server**:

In `index.js`, set up a basic server:

```
const express = require('express');
const app = express();
const PORT = 3000;

app.use(express.json());

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

2. Project Structure

Here's the recommended directory structure for your API:

```
ToDoListAPI/
│
├── index.js        # Entry point
├── routes/
│   └── todos.js    # Contains routes for CRUD operations
└── models/
    └── todo.js     # Data model for a to-do item (optional for simplicity)
```

3. API Routes and Implementation

Overview of Routes

Method	Route	Description
POST	/todos	Create a new to-do
GET	/todos	Retrieve all to-dos
GET	/todos/:id	Retrieve a specific to-do by ID
PUT	/todos/:id	Update a specific to-do by ID
DELETE	/todos/:id	Delete a specific to-do by ID

Route Details

1. `POST /todos` - Create a New To-Do

- **Request**:

Body (JSON format):

```
{
  "title": "Sample to-do item",
  "description": "Description of the to-do item"
}
```

- **Response**:

201 Created:
json

```
{
  "id": 1,
  "title": "Sample to-do item",
  "description": "Description of the to-do item",
  "completed": false
}
```

- **Error Handling**:
  - 400 Bad Request if `title` is missing.

2. `GET /todos` - Retrieve All To-Dos

- **Response**:

200 OK:
Json

```
[
  {
    "id": 1,
    "title": "Sample to-do item",
    "description": "Description of the to-do item",
```

```
    "completed": false
  }
]
```

- **Error Handling**:
  - 500 Internal Server Error for unexpected issues.

3. `GET /todos/:id` - Retrieve a To-Do by ID

- **Response**:

200 OK:
json

```
{
  "id": 1,
  "title": "Sample to-do item",
  "description": "Description of the to-do item",
  "completed": false
}
```

- **Error Handling**:
  - 404 Not Found if the ID does not exist.

4. `PUT /todos/:id` - Update a To-Do by ID

- **Request**:

Body (JSON format):
json

```
{
  "title": "Updated to-do item",
  "description": "Updated description",
  "completed": true
}
```

- **Response**:

200 OK:
json

```
{
  "id": 1,
  "title": "Updated to-do item",
  "description": "Updated description",
  "completed": true
}
```

- **Error Handling**:
  - 404 Not Found if the ID does not exist.
  - 400 Bad Request if `title` is missing.

5. `DELETE /todos/:id` - Delete a To-Do by ID

- **Response**:

200 OK:
json

```
{
  "message": "To-do item deleted successfully"
}
```

- **Error Handling**:
  - 404 Not Found if the ID does not exist.

4. Error Handling Guidelines

To handle common errors effectively:

- For **missing fields** (e.g., `title` in POST requests), respond with `400 Bad Request` and a message indicating the missing field.

- For **invalid or non-existing IDs** in GET, PUT, and DELETE requests, respond with `404 Not Found`.
- Use `500 Internal Server Error` for any unexpected issues, such as server errors.

5. Testing Your API

Use a tool like **Postman** test each route:

- **Postman**: Set up requests for each route and verify the responses.

Sample Responses and Tips

- **Ensure consistent JSON responses**: Always return JSON with meaningful fields.
- **Testing edge cases**: Try adding to-dos without required fields, or updating/deleting a to-do with an invalid ID, to see how your API handles errors.

By following this documentation, you'll be able to build a fully functional To-Do List API and gain experience with Node.js, Express, and RESTful APIs. Good luck!