**Task Manager API and Database Development Requirements**

**Overview:**

We need to develop a set of APIs for a task manager application built with Angular. These APIs will perform CRUD (Create, Read, Update, Delete) operations on tasks and will interact with a database to store and manage task data. Below are the detailed requirements for the APIs and the database schema.

**API Endpoints:**

1. **Create New Task**
   * **Endpoint:** POST /api/tasks
   * **Description:** Creates a new task with a title, description, and status.
   * **Request Body:**

json

Copy code

{

"title": "Task Title",

"description": "Task Description",

"status": "Task Status"

}

* + **Response:**

json

Copy code

{

"message": "Task created successfully",

"task": {

"id": "Unique Task ID",

"title": "Task Title",

"description": "Task Description",

"status": "Task Status"

}

}

1. **Update Task Status**
   * **Endpoint:** PUT /api/tasks/:id/status
   * **Description:** Updates the status of an existing task.
   * **Request Body:**

json

Copy code

{

"status": "Updated Task Status"

}

* + **Response:**

json

Copy code

{

"message": "Task status updated successfully",

"task": {

"id": "Task ID",

"status": "Updated Task Status"

}

}

1. **Edit Task**
   * **Endpoint:** PUT /api/tasks/:id
   * **Description:** Updates the title, description, and status of an existing task.
   * **Request Body:**

json

Copy code

{

"title": "Updated Task Title",

"description": "Updated Task Description",

"status": "Updated Task Status"

}

* + **Response:**

json

Copy code

{

"message": "Task updated successfully",

"task": {

"id": "Task ID",

"title": "Updated Task Title",

"description": "Updated Task Description",

"status": "Updated Task Status"

}

}

1. **Delete Task**
   * **Endpoint:** DELETE /api/tasks/:id
   * **Description:** Deletes an existing task.
   * **Response:**

json

Copy code

{

"message": "Task deleted successfully"

}

**Database Schema:**

* **Table Name:** tasks
* **Columns:**
  + id (Primary Key): A unique identifier for each task.
  + title: The title of the task.
  + description: A detailed description of the task.
  + status: The current status of the task (e.g., "pending", "in-progress", "completed").

**Example Table Structure:**

sql

Copy code

CREATE TABLE tasks (

id SERIAL PRIMARY KEY,

title VARCHAR(255) NOT NULL,

description TEXT,

status VARCHAR(50) NOT NULL

);

**Implementation Notes:**

* Use RESTful conventions for the API endpoints.
* Ensure proper validation and error handling in the APIs.
* Use appropriate HTTP status codes for responses.
* Secure the API endpoints as needed (e.g., using authentication/authorization mechanisms).
* Consider using a framework like Express.js for setting up the API server.
* For the database, PostgreSQL or MySQL can be used based on preference.

This detailed description should provide the developer with a clear understanding of the requirements and facilitate the implementation of the APIs and the database.